Enterprise Systems Engineering (ESE)
1400 Concord Point Lane
Reston, VA 20194
Phone: (703) 478-0166
Fax: (703) 478-6299

National Oceanic and Atmospheric Administration (NOAA)

Commitment Tracking Interface
Detailed Design Document - Final

April 11, 2002

CAMS

A System For Now
And The Future

Table of Contents

List of Figures

1. Introduction

1.1 Background and Objectives

The Department of Commerce (DOC), National Oceanic and Atmospheric Administration (NOAA) plans to fully implement the Commerce Administrative Management System (CAMS) by the beginning of Fiscal Year 2003. A major component of the CAMS is the Core Financial System (CFS). The CFS was selected as the standard accounting system for the Department of Commerce bureaus. CFS supports:

- Core Financial System Management,
- General Ledger Management,
- Funds Management,
- Payment Management,
- Cost Management,
- Receipt Management, and
- Reporting.

Once implemented, CAMS will serve as the accounting system of record for NOAA. At the point CAMS is fully implemented, the existing system of record, NOAA's Financial Management System (FIMA), will be decommissioned. Therefore, any data obtained from FIMA or Interactive FIMA (IFIMA) to support interfacing systems will not be available. The objective of the Commitment Tracking Interface project is to ensure that the Line and Staff Offices of NOAA can continue to receive the same or similar information/data to what is currently received from FIMA/IFIMA to support the existing Line Office commitment tracking/Management Information System (MIS) systems.

The requirements of the project are to produce the following deliverables:

- Develop a Requirements Document that details NOAA's needs specifically for creating an interface to provide CAMS data to Commitment Tracking/MIS systems.
- Prepare a Detailed Design document to define interface specifications, flat file layouts, high level program logic, temporary table definitions, and log report definitions required for the development effort.

- Develop programs and associated scripts and data definition language to implement the approved design.
- Develop a test plan to be used by the Government's software acceptance team to exercise and demonstrate the expected features of the data warehouse.
- Guide and support the acceptance testing process, review results, and make revisions as necessary.
- Provide a System Operations Guide for the interface to detail the interface operations, reports, and user interface.

1.2 Scope

The scope of this project is to develop and provide: a requirements specification, design document, test scripts, operations guide, operational code, and scripts for a standard CAMS interface to provide "commitment tracking" data to NOAA commitment tracking/MIS systems.

Note that the term "commitment" in this document refers to transactions and/or financial events that are in the "pipeline"; including Budget Operating Plans, commitments, obligations, accruals, reservations, cost adjustments, and disbursements/expenses.

The Commitment Tracking Interface shall provide data to the Line Office's existing systems (which will require system changes to become "CAMS" data oriented). It will support an interface (via Oracle tables or flat files) to NOAA's Line Office commitment tracking/ MIS systems from CAMS' data warehouse/data marts. This interface's goal is to ensure that the data being interfaced provides the same or similar capability as that currently provided.

"Same" data would be that data for which there is an exact match of values (for example, the first four digits of the object class code).

"Similar" data is that which serves the same basic purpose but which may be structured differently (for example, the CAMS Project Code [38R1BA7] is similar to the FIMA Task Code [8R1BA7] but adds a leading digit).

It is also important to note that the NOAA Data Mart presents/summarizes information from CAMS/CFS. It does NOT create new information.

Due to the time period in which to implement this capability, the requirements for this project have been prioritized as follows:

Priority 1:

- Budget Operating Plan (BOP) Data
- Obligations (CAMS SGL 4800 Series)
- Expenses (CAMS SGL 4900 Series)
- Cost Adjustments (Corrections to Accounting on Vendor Invoices [PM006])
- Receipt of Goods
- Labor – Summary Level

Priority 2:

- Labor – Detail Level
- Feeder System Reservations (i.e., Travel Authorizations)

Priority 3 or Later:

- Commitments (CAMS SGL 4700)
- Summary Level Transfers
- Commerce Purchase Card System (CPCS) Log Orders (Requires CSC agreement to make application changes to CPCS.)

Note that the Priority 2/3 items will NOT be available in the initial June 2002 delivery of the NOAA Data Mart. However, the priority 2 items should be available for September 30, 2002.

This task is NOT to build Line Office custom reports or queries, but rather to ensure that the Line Offices can obtain the CAMS data they require.

1.3 References

This document was compiled through the conduct of user interviews and review of supporting technical documentation. The following individuals were consulted and provided input to this document:

1. Alexander, Vicky, Department of Commerce, NOAA, National Weather Service, Central Region.
2. Barnes, Lillian, Department of Commerce, NOAA, Office of Finance and Administration.

3. Bass, Gregory, Department of Commerce, NOAA, Office of Marine and Aviation Operations.

4. Boller Mike, Department of Commerce, NOAA, Office of Finance and Administration.

5. Brown, Agnes, Department of Commerce, NOAA, National Weather Service, Headquarters/CFO.

6. Brown, Allen, Department of Commerce, NOAA, Office of Finance and Administration.

7. Brown, Annette, Department of Commerce, NOAA.

8. Burger, Eugene, Department of Commerce, NOAA, Office of Oceanic and Atmospheric Research.

9. Cartwright, Chris, Department of Commerce, NOAA, National Environmental Satellite and Data Information Service.

10. Coleman, Lois, Department of Commerce, NOAA, Office of Finance and Administration.

11. Cook, Rhonda, Department of Commerce, NOAA, National Weather Service, Central Region.

12. Dominic, R. J., Department of Commerce, NOAA, Office of Finance and Administration.

13. Driscoll, David, Department of Commerce, NOAA, National Marine Fisheries Service (NMFS).

14. Foster, Violet, Department of Commerce, NOAA, National Weather Service.

15. Hay, Cindee, Department of Commerce, NOAA, National Weather Service, Central Region.

16. Hay, Rhonda, Department of Commerce, NOAA, National Weather Service, Central Region.

17. Heaton, Glen, Department of Commerce, NOAA, National Weather Service, Southern Region.

18. Henderson, Robert, Department of Commerce, NOAA, Office of Finance and Administration.

19. Hendsbee, David, Department of Commerce, NOAA, National Ocean Service.

20. Hodges, Lynn, Department of Commerce, NOAA, National Weather Service, Office of Operational Systems.

21. Holdsworth, William, Department of Commerce, NOAA, Office of Finance and Administration.

22. Hughes, Pamela, Department of Commerce, NOAA, National Environmental Satellite and Data Information Service.

23. Ingels, Millie, Department of Commerce, NOAA, Office of Finance and Administration.

24. Lewis, Diana, Department of Commerce, NOAA, Office of Marine and Aviation Operations.

25. Loitsch, Cindy, Department of Commerce, NOAA, Office of Oceanic and Atmospheric Research.

26. Lord, John, Department of Commerce, NOAA, National Weather Service.

27. Lovett, Ann, Department of Commerce, NOAA, Office of Finance and Administration.

28. Marrazzo, Carolyn, Department of Commerce, NOAA, Office of Finance and Administration.

29. Marth, Donnie, Department of Commerce, NOAA, National Ocean Service.

30. Martin, Gerry, Department of Commerce, NOAA, National Weather Service, National Data Buoy Center.

31. McNeary, Caroline, Department of Commerce, NOAA, National Weather Service, OCWWS.

32. Nguyen, Cristina, Department of Commerce, NOAA, Office of Finance and Administration.

33. O'Connor, John, Department of Commerce, NOAA, National Weather Service, Headquarters/CFO.

34. O'Connor, Pat, Department of Commerce, NOAA, Office of Finance and Administration.

35. Olivere, Peter, Department of Commerce, NOAA, Office of Finance and Administration.

36. Parr, Eric, Department of Commerce, NOAA, National Weather Service.

37. Price, Gregory, Department of Commerce, NOAA, Office of Finance and Administration.

38. Pulver, Maureen, Department of Commerce, NOAA, National Marine Fisheries Service (NMFS).

39. Rieck, Mary, Department of Commerce, NOAA, National Weather Service.

40. Rubio, Linda, Department of Commerce, NOAA, Office of Marine and Aviation Operations.

41. Ryan, Sandra, Department of Commerce, NOAA, Office of Finance and Administration.

42. Schornick, Glenda, Department of Commerce, NOAA, National Weather Service, National Data Buoy Center.

43. Semones, Rica, Department of Commerce, NOAA, Office of Oceanic and Atmospheric Research.

44. Shultz, Sue, Department of Commerce, NOAA, National Weather Service, Central Region.

45. Sparks, Larry, Department of Commerce, NOAA, Office of Finance and Administration.

46. Stark, Gerald, Department of Commerce, NOAA, National Marine Fisheries Service (NMFS).

47. Stasulli, Roxanne, Department of Commerce, NOAA, National Weather Service.

48. St. Clair, Karen, Department of Commerce, NOAA, National Weather Service, Western Region.

49. Williams, Terri, Department of Commerce, NOAA, National Environmental Satellite and Data Information Service.

50. Woods, Lenora, Department of Commerce, NOAA, Office of Finance and Administration.

51. Yoder, Jaye, Department of Commerce, NOAA, National Weather Service, Management & Finance.

52. Zuckerberg, Lisa, Department of Commerce, NOAA, Office of Marine and Aviation Operations.

The following documents were used as reference material in compiling this document:

1. Accounting Transactions Edit and Update Program Requirements, Office of Comptroller, Financial Management Division, Financial Systems Branch, June 1993.

2. Financial Analysis and Commitment Tracking System (FACTS) Procedures Manual, Version 5.0, September 2000, Larry Sparks, Chief, Systems Development Branch, Systems Division, Information Systems Office, Office of Financial Administration, National Oceanic and Atmospheric Administration.

3. WASC Budget Tracking System (WBTS).

4. FACTS for the NOS CAMS Pilot.

5. Labor (LAB) Interface Technical Documentation, Version 1.0, September 9, 2000.

2. Environment

The Department of Commerce is developing software to create a CAMS data warehouse. The CAMS data warehouse will provide detailed trial data (by General Ledger account) and summarized data using common Department of Commerce definitions for all accounting events. This data warehouse will provide a common capability for all CAMS bureaus (e.g., NOAA, Census, NIST, etc.) and each bureau will install the DOC-provided software and load their data in their data warehouse.

Data for the Commitment Tracking interface will be provided through a data mart. A data mart is a focused subset of information from the data warehouse that addresses a specific requirement(s). CAMS will have two types of data marts:

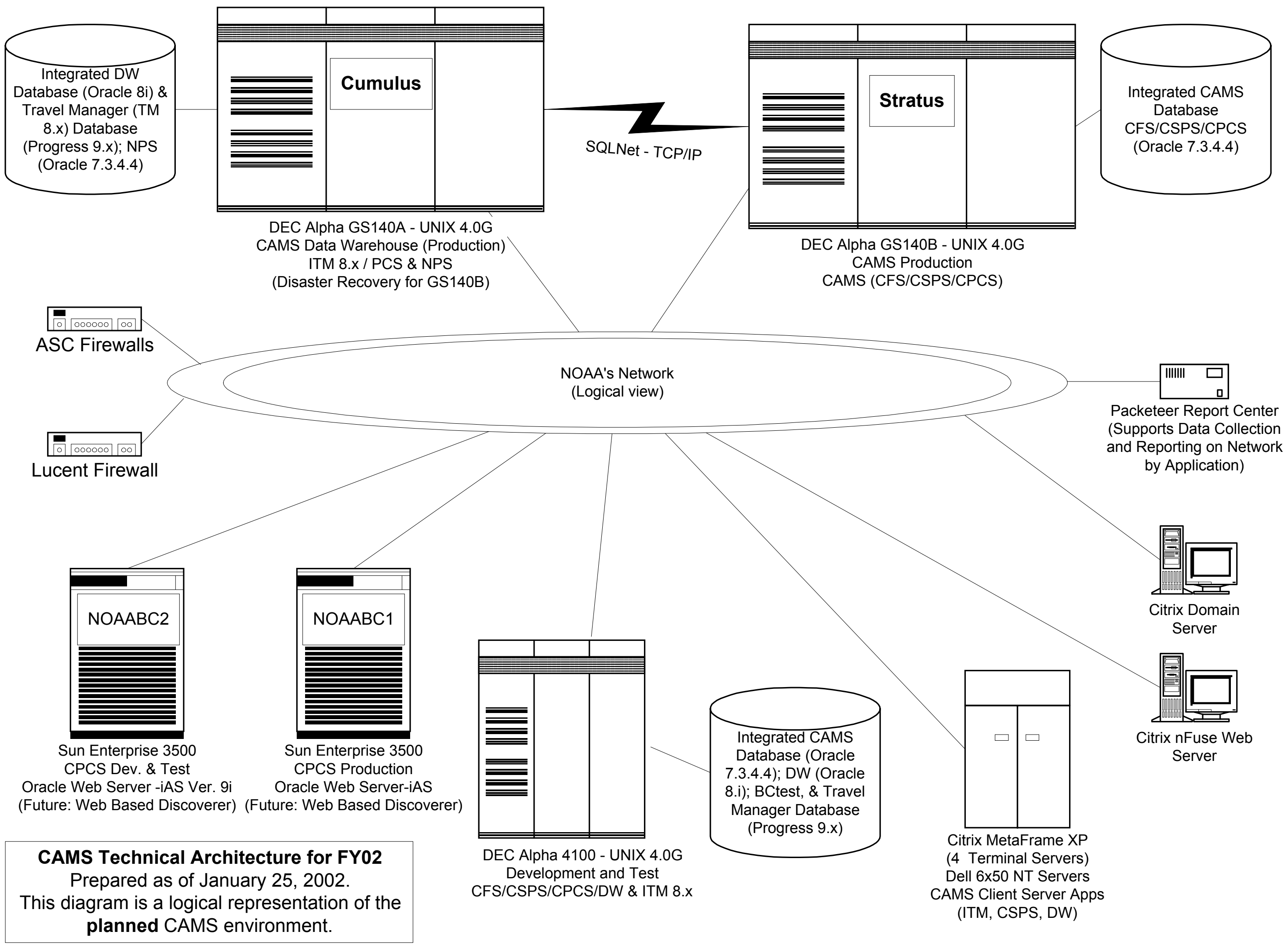
- Department of Commerce provided software – These data marts employ common Department of Commerce-wide definitions (i.e., Budget, etc.) and are populated by standard software.
- NOAA/BXA-Specific – These data marts are developed specifically to support NOAA's business needs (i.e., Integrated Travel Manager [ITM] Travel Manager Travel Authorizations [TMTA], Commerce Purchase Card System [CPCS], Pilot, Accounts Payable Payment Inquiry, etc.). Currently, the ITM TMTA and CPCS data marts exist.

When fully implemented, the CAMS data warehouse and data marts installed at NOAA will reside on the Alpha GS140A server (Cumulus). The CAMS production system (Core Financial System, Commerce Small Purchase System [CSPS], and the Commerce Purchase Card System [CPCS]) for transaction processing will be hosted on the Alpha GS140B server (Stratus). Other aspects of the CAMS environment include:

- CAMS is implemented using the Oracle Relational Database Management System (RDBMS) and supporting development software.
- NOAA currently has a NOAA-wide license for the Oracle RDBMS, Forms, Reports, and Web for CAMS applications.
- SQL*Plus licenses are NOT included.
- NOAA also has licenses for Oracle Discoverer, a data warehouse/data mart Graphical User Interface (GUI) query tool.

- The NOAA CAMS office will provide the end-user (client-side) license for Oracle Discoverer. However, each line office will be responsible for user training of this tool.
- Discoverer also has a server that requires set-up for access.
- There will be no Open Database Connectivity (ODBC) access to CFS.
- The data warehouse and data marts will be accessed via menus to obtain on-line canned queries and reports.
- Each Line Office may have custom queries developed or develop their own through Discoverer.
- The ability to perform batch extracts will also be provided.

Figure 1 provides a logical representation of the planned CAMS operating environment.


Integrated DW Database (Oracle 8i) & Travel Manager (TM 8.x) Database (Progress 9.x); NPS (Oracle 7.3.4.4)
Cumulus
DEC Alpha GS140A - UNIX 4.0G
CAMS Data Warehouse (Production)
ITM 8.x / PCS & NPS
(Disaster Recovery for GS140B)
SQLNet - TCP/IP
Stratus
DEC Alpha GS140B - UNIX 4.0G
CAMS Production
CAMS (CFS/CSPS/CPCS)
Integrated CAMS Database CFS/CSPS/CPCS (Oracle 7.3.4.4)
ASC Firewalls
Lucent Firewall
NOAA's Network
(Logical view)
Packeteer Report Center
(Supports Data Collection and Reporting on Network by Application)
Citrix Domain Server
Citrix nFuse Web Server
NOAABC2
NOAABC1
Sun Enterprise 3500
CPCS Dev. & Test
Oracle Web Server -iAS Ver. 9i
(Future: Web Based Discoverer)
Sun Enterprise 3500
CPCS Production
Oracle Web Server-iAS
(Future: Web Based Discoverer)
DEC Alpha 4100 - UNIX 4.0G
Development and Test
CFS/CSPS/CPCS/DW & ITM 8.x
Integrated CAMS Database (Oracle 7.3.4.4); DW (Oracle 8.i); BCtest, & Travel Manager Database (Progress 9.x)
Citrix MetaFrame XP
(4 Terminal Servers)
Dell 6x50 NT Servers
CAMS Client Server Apps
(ITM, CSPS, DW)

CAMS Technical Architecture for FY02
Prepared as of January 25, 2002.
This diagram is a logical representation of the **planned** CAMS environment.

Figure 1. Logical Representation of Planned CAMS Operating Environment

3. High Level Design

The initial design of the NOAA Data Mart is to provide a table structure that allows Line/Field Offices the ability to extract “same or similar” data for their commitment tracking/MIS systems that they currently get from IFIMA.

This section describes the overall strategy for:

1. Extracting the data from the production Core Financial System (CFS)

2. Populating the NOAA Data Mart Summary tables for Budget Operating Plan data, data having a general ledger impact by Standard General Ledger Account, and data having a general ledger impact by NOAA-defined financial categories.

3. Populating transaction level detail tables for commitments (priority 3), obligations, expenses, labor, Budget Operating Plans, and reservations (transactions that have no general ledger effect).

4. Managing the NOAA Data Mart refresh process to ensure successful completion and data integrity.

5. Data security and access.

6. Line/Field Office extract processing.

This section will also provide a mapping of the IFIMA tables used by the Line/Field Offices to the NOAA Data Mart tables.

3.1 NOAA Data Mart Refresh Process

The NOAA Data Mart will be populated based on CFS tables currently being extracted by the CAMS data warehouse effort along with additional table data required for the NOAA Data Mart. Figure 2 illustrates the overall strategy of how the NOAA Data Mart will be populated.

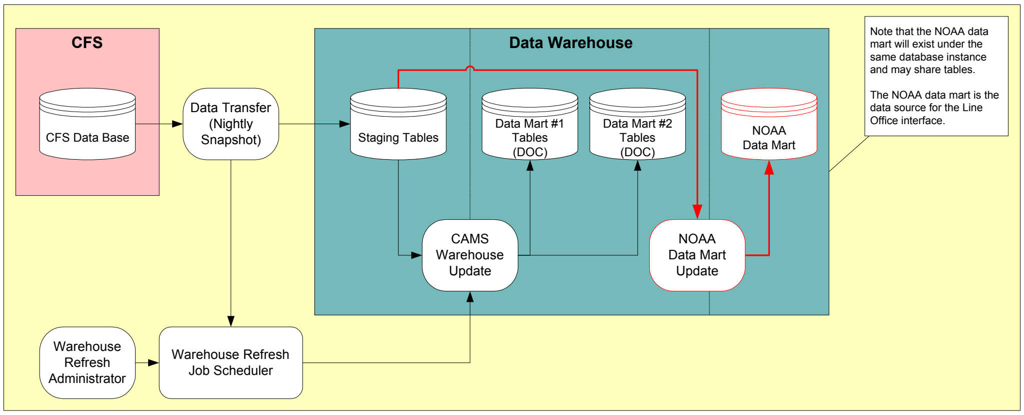

CFS
CFS Data Base
Data Transfer (Nightly Snapshot)
Data Warehouse
Staging Tables
Data Mart #1 Tables (DOC)
Data Mart #2 Tables (DOC)
NOAA Data Mart
CAMS Warehouse Update
NOAA Data Mart Update
Warehouse Refresh Administrator
Warehouse Refresh Job Scheduler
Note that the NOAA data mart will exist under the same database instance and may share tables.
The NOAA data mart is the data source for the Line Office interface.

Figure 2. NOAA Data Mart Extract Process

The CAMS Data Warehouse refresh process is run on a nightly basis. This routine takes a delta snapshot of the core CFS tables that it places in a staging area. From this staging area, it then populates its various data marts.

As part of the CAMS Data Warehouse process that takes a snapshot of CFS tables, NOAA will introduce several additional tables required for populating NOAA Data Mart tables and for tracking data status. These additional tables are:

1. **APERIOD** – Provides information concerning the status of the various accounting periods (i.e., OPEN, PRELIMINARY CLOSE, or CLOSED).

2. **CBS_TRANS_ITEM_D** – Information specific to Commerce Purchase Card System (CPCS) transactions. From this table the merchant name and reference number can be provided.

3. **CBS_TRANSACTIONS_D** – Information specific to CPCS transactions. From this table the cardholder’s number for this transaction can be provided.

4. **CBS_CARD_HOLDER_L** - Information specific to CPCS transactions. From this table the ASC number and employee number (from which the cardholder’s name can be derived) can be provided.

5. **CBS_NOTES_D** – First entry for a transaction should contain the description of what was procured.

6. **GJ_CONTROL** – For labor document level information.

7. **GJ_DETAIL** – For summarized labor data to support providing labor data at the employee level.

8. **GJ_EMPLOYEE** – For detailed employee data.

9. **EMPLOYEE_CONTROL** – Employee data.

Once CAMS Data Warehouse completes the snapshot portion of the refresh, the NOAA Data Mart may begin its refresh process. Figure 3 presents the routines associated with the NOAA Data Mart refresh process and the sequence in which they would run. Each of these processes is described in detail in Section 4, Detailed Design.

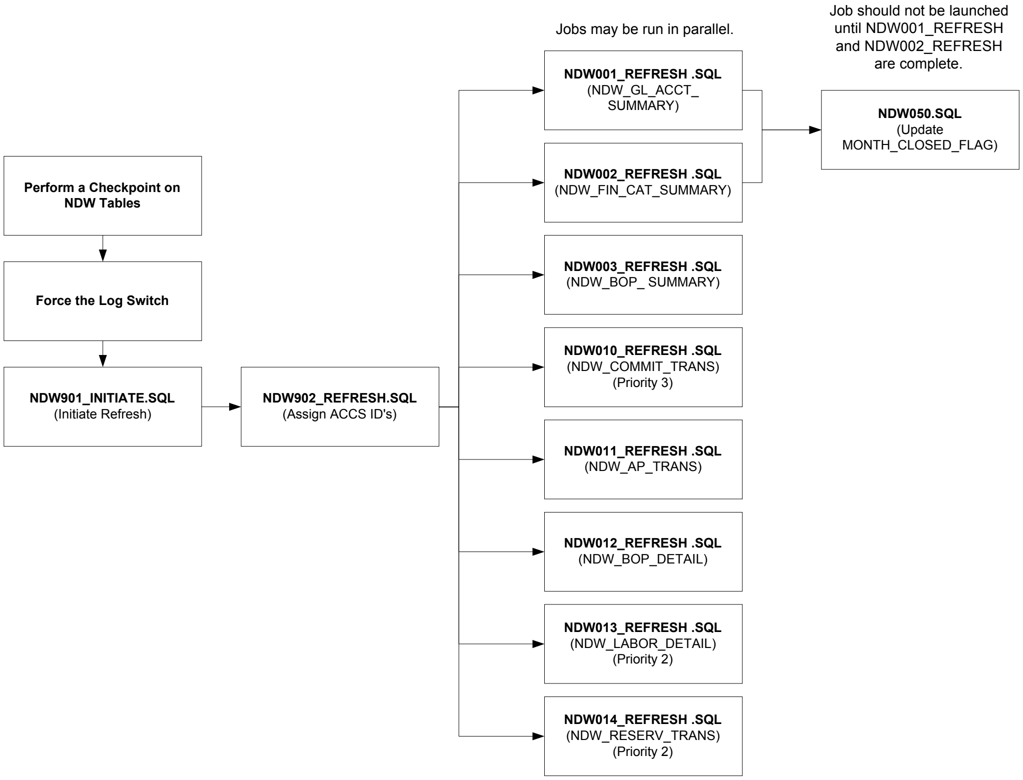

Jobs may be run in parallel.
Job should not be launched until NDW001_REFRESH and NDW002_REFRESH are complete.
Perform a Checkpoint on NDW Tables
Force the Log Switch
NDW901_INITIATE.SQL (Initiate Refresh)
NDW902_REFRESH.SQL (Assign ACCS ID's)
NDW001_REFRESH .SQL (NDW_GL_ACCT_ SUMMARY)
NDW002_REFRESH .SQL (NDW_FIN_CAT_SUMMARY)
NDW003_REFRESH .SQL (NDW_BOP_ SUMMARY)
NDW010_REFRESH .SQL (NDW_COMMIT_TRANS) (Priority 3)
NDW011_REFRESH .SQL (NDW_AP_TRANS)
NDW012_REFRESH .SQL (NDW_BOP_DETAIL)
NDW013_REFRESH .SQL (NDW_LABOR_DETAIL) (Priority 2)
NDW014_REFRESH .SQL (NDW_RESERV_TRANS) (Priority 2)
NDW050.SQL (Update MONTH_CLOSED_FLAG)

Figure 3. NOAA Data Refresh Process Flow Sequence

Note that there are several business rules that will be enforced through the code for performing a NOAA Data Mart refresh:

1. Refresh jobs may need to be run multiple times within the same day.

2. For every table updated, the refresh routine must track the starting and ending transaction numbers that were processed.

3. Each refresh job must produce a log file to indicate whether the job was successfully started or not (e.g., if the Oracle database instance was available or not) along with the database log table entries. A separate Oracle database log table must be

maintained to record when the job started and finished, and interim status lines.

4. Each batch job must provide feedback to the operator on the status of the job with interim messages sent to the terminal reporting progress (e.g., the message "NDW001 now processing TRIAL_ID 9999999" would be displayed for every 100th record processed).

5. All jobs must use standard error messages from the CAMS RETURN_CODE_CONTROL table using existing codes and messages to the maximum extent possible. At a minimum, any displayed or written system messages should indicate the job name, the date and time, the transaction number or relevant key fields, and specific information identifying the problem to include field names and values where possible.

6. All code should be written to correct any issues encountered through a back out of invalid data and/or a refresh versus a complete table/data mart reload.

7. Code should be written to allow parallel processing of jobs against single tables (for example, if two routines need to add items to the same table, the code should be written to allow this … i.e., no exclusive writes).

8. A refresh should be able to be stopped in the middle of the process by the operator in a controlled manner (versus being aborted) and then resumed at a later time continuing from where it left off.

9. To ensure tables within the NOAA data mart balance with one another, only one NDW refresh process should be running at a time. Although processes within a single refresh can be run in parallel, a new refresh should not be started until the previous one completes.

10. A new refresh should not be allowed to be initiated if the previous one did not process through completion.

11. Refresh routines should be able to be run independently of each other should complete table re-builds be necessary for specific tables.

3.2 NOAA Data Mart Table Structure

The NOAA Data Mart will provide a series of tables for Line/Field Office extraction into their commitment tracking/MIS systems. Figure 4 identifies the main tables that comprise the NOAA Data Mart. In addition to these, there are a series of support tables that are described in Section 3.2.3, NOAA Data Mart Support Table Definitions with the table formats provided in Appendix C – Support Table Record Layouts.

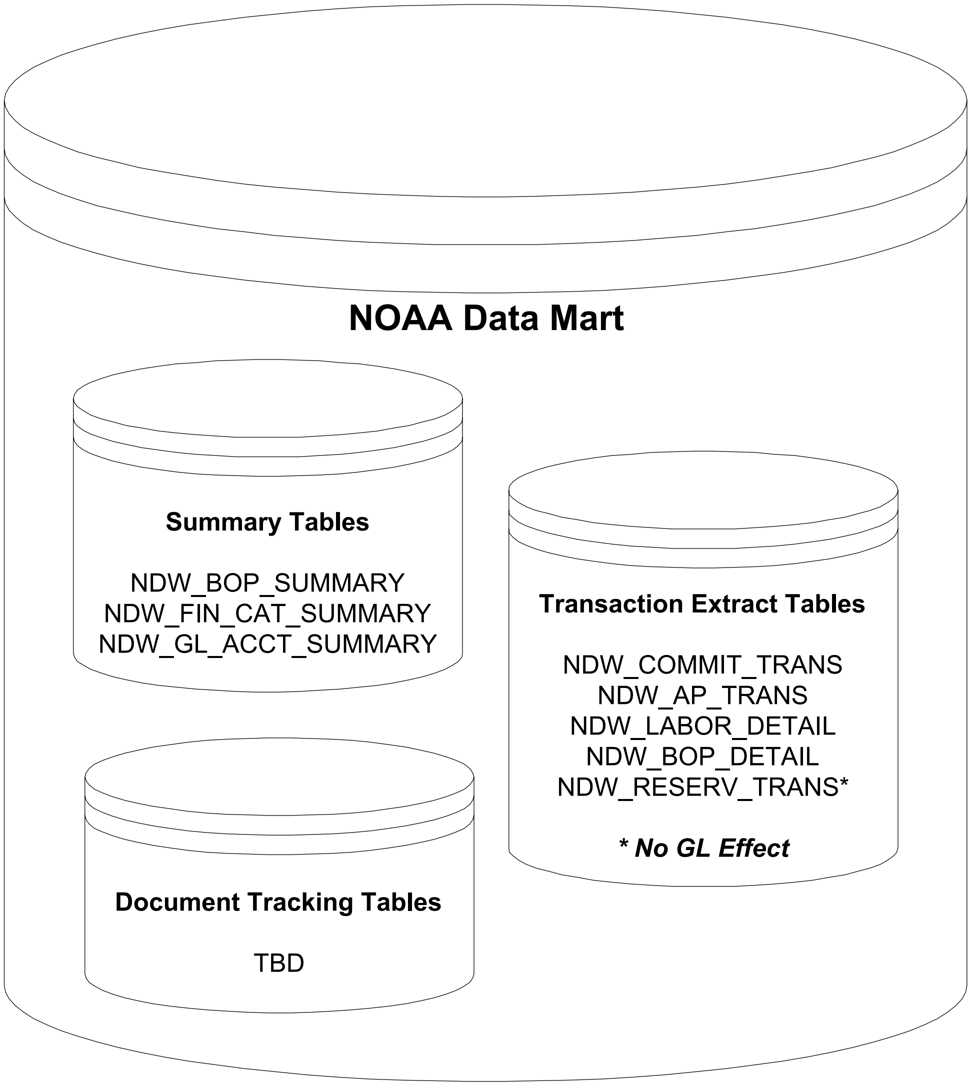


Figure 4. NOAA Data Mart Tables

The tables are grouped by whether they are Summary Tables that provide data summarized by ACCS and accounting period (Fiscal Month and Fiscal Year) and Transaction Extract Tables that provide detailed records of every transaction that either incremented an accounting amount or decremented an amount. For example, for a single line item from a vendor invoice that was initiated through a Purchase Order, two transactions will be captured (both in NDW_AP_TRANS) in the NOAA Data Mart, an unpaid expense record showing the amount debited to this account and a record to liquidate the amount (as a credit) from the undelivered order that was posted for that Purchase Order line item. In essence, this transaction takes the amount out of the undelivered order category and places it in the unpaid expense category. The third group of tables, Document Tracking Tables, are not directly required for the extraction of records, but will be created as a lower priority item to assist in the analysis of transactions and the reconciliation of charges.

Figure 5 illustrates the mapping of the IFIMA tables currently being used by the Line/Field Offices as the source of data for their commitment tracking/MIS systems and how they map to the NOAA Data Mart tables.

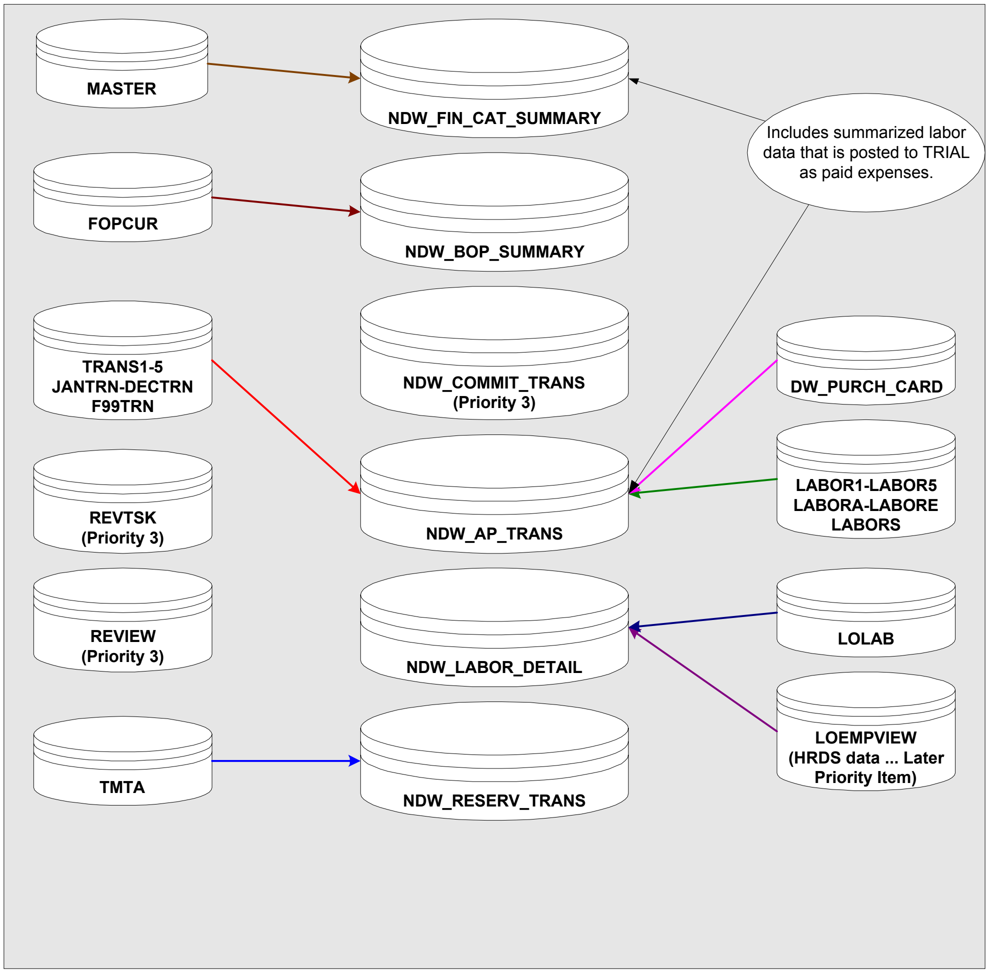


Figure 5. Mapping of IFIMA Tables to NOAA Data Mart Tables

The NOAA Data Mart will provide “same or similar” data to what is stored in IFIMA. Some of the differences in data include:

1. Any data currently available in IFIMA that is generated outside of CAMS and will not be converted, and therefore will not be available through the NOAA Data Mart. For example, personnel data.

2. The IFIMA TRANS1-5 tables group transactions on a weekly basis. CAMS does not process with a weekly concept. Data will be downloaded on a nightly basis into the NOAA Data Mart and will be summarized at the monthly level.

3. CAMS does not maintain the concept of a single document number that is unique across all document types. Each document type is generally controlled by a unique transaction number. The value of which may be the same across document types (e.g., there can exist a Purchase Order with a transaction number equal to 1, a vendor invoice with a transaction number equal to 1, a Budget Operating Plan with a transaction number equal to 1 and they can all be unrelated).

   The detailed transactions will capture all transaction numbers relevant at the time the transaction was generated providing a variety of ways in which to query the data.

3.2.1 NOAA Summary Tables

The CFS TRIAL table is used as the basis of reliable information for recording general ledger accounting events. For each transaction processed in CFS, multiple transactions showing the movement of money are posted to the TRIAL table. Although this table provides a wealth of information, the size of the table is too large for efficient processing (as of March 2002, there were over 30 million rows) and much of the data is encoded and hard for the general user to interpret. Each transaction is given a Standard General Ledger account number that identifies the type of transaction it is. There are 360 active account/sub-account numbers defined within CFS of which only 25 are of interest to this effort.

For both reporting and extract purposes, the NOAA Data Mart will summarize this data at two levels as shown in Figure 6.

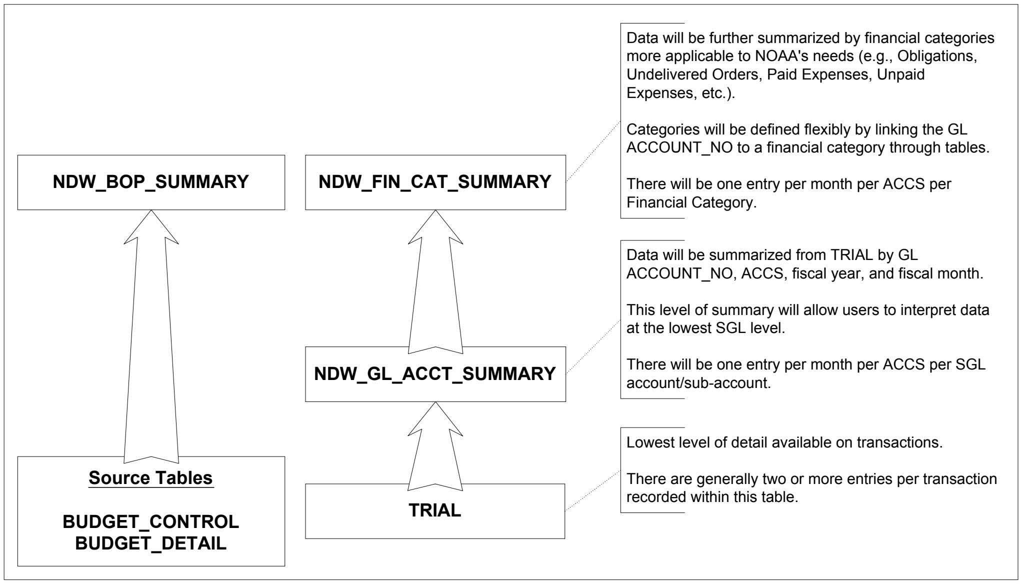

NDW_BOP_SUMMARY
NDW_FIN_CAT_SUMMARY
NDW_GL_ACCT_SUMMARY
Source Tables
BUDGET_CONTROL
BUDGET_DETAIL
TRIAL
Data will be further summarized by financial categories more applicable to NOAA's needs (e.g., Obligations, Undelivered Orders, Paid Expenses, Unpaid Expenses, etc.).
Categories will be defined flexibly by linking the GL ACCOUNT_NO to a financial category through tables.
There will be one entry per month per ACCS per Financial Category.
Data will be summarized from TRIAL by GL ACCOUNT_NO, ACCS, fiscal year, and fiscal month.
This level of summary will allow users to interpret data at the lowest SGL level.
There will be one entry per month per ACCS per SGL account/sub-account.
Lowest level of detail available on transactions.
There are generally two or more entries per transaction recorded within this table.

Figure 6. NOAA Data Mart Summary Tables

The first level is by standard general ledger account/sub-account number, month and fiscal year (based on general ledger accounting period), and ACCS. This groups the data at the lowest standard general ledger level for all account/sub-account numbers that can be generated through CFS – not just those required by the Line/Field Office commitment tracking interfaces. Therefore, not only will undelivered order data be available, but also accounts receivable, liabilities, appropriations, etc. There will be a row entered for each account/sub-account number, fiscal year, fund code fiscal year, month, and unique ACCS (does not include the user defined portion of the ACCS). This summarization is stored in the NDW_GL_ACCT_SUMMARY table.

The second level is at one level higher, where one or more standard general ledger accounts are combined into a NOAA-defined category. Categories are defined or can be changed at the beginning of a fiscal year. Unlike the NDW_GL_ACCT_SUMMARY where there is one row for each standard general ledger account/sub-account number, the NDW_FIN_CAT_SUMMARY shows the values for all defined categories for a month, fiscal year, fund code fiscal year, ACCS combination. Therefore, only a single row needs to be read to obtain the amounts for all categories for a unique ACCS for the month.

In addition to summarizing data from the TRIAL table, the NOAA Data Mart will also provide summarized Budget Operating Plan data in the NDW_BOP_SUMMARY table. This table groups the BOP data by month, fiscal year and ACCS. Note that, unlike TRIAL transactions, some BOPs do not use the entire ACCS. For example, a BOP may be defined at the program level. In this case fields such as the Project Code and Task Code would be stored as all zeroes.

Note that summary tables summarize data on a daily basis at the general ledger account period level (month). During an open general ledger account period, the amounts in the summary tables will change to reflect the new transactions processed. They will not reflect the delta amounts from day to day, only the total values.

The formats for the NOAA Data Mart summary tables are provided in Appendix A – Summary Table Record Formats.

3.2.2 NOAA Transaction Extract Tables

The source of detailed transaction data for the Line/Field Offices will be found in the NOAA transaction extract tables. These tables will be generated from TRIAL entries and are organized by type of data:

- **NDW_COMMIT_TRANS** – Transactions with an account number equal to 4700. NOAA currently does not do commitment based accounting; therefore, this table will be implemented as a later phase item.

- **NDW_AP_TRANS** – Transactions with an account number between 4800 and 4899 (obligations) and transactions with an account number between 4900 and 4999 (expenses).

- **NDW_LABOR_DETAIL** – Detailed labor transactions (by employee) that are provided by the NFC and are stored in CFS in the GJ_EMPLOYEE table. These entries are also summarized by the NOAA NFC interface and posted to CFS as summarized entries under the paid expenses account number.

- **NDW_BOP_DETAIL** – Detailed Budget Operating Plan data from entries made in the Budget Operating Plan Transaction screen (FM066).

- **NDW_RESERV_TRANS** – Within the NOAA Data Mart, reservations are defined as transactions that are generated outside of the Core Financial System that may eventually result in a commitment, estimated accrual, obligation, or expense; however, they do not have a general ledger effect within CFS. The existing Travel Manager Travel Authorization (TMTA) data mart will be converted over to this table format. If the decision is made to record accounting with Commerce Purchase Card System (CPCS) log orders, that information may also be considered a reservation.

The two primary transaction tables generated from the CFS TRIAL table (NDW_COMMIT_TRANS and NDW_AP_TRANS) will be comprised of transactions that generate the entry followed by the transaction(s) that liquidates the entry. For example, a purchase order is approved for $50 resulting in an undelivered order entry in the NDW_AP_TRANS table. The vendor invoice is received and approved that liquidates the undelivered order (entry in NDW_AP_TRANS for -$50) and is recorded as an unpaid expense (entry in NDW_AP_TRANS for $50). These tables will only have entries inserted into them – not modified or deleted.

Note that transactions are recorded at the document/item/line (also called Multiple Distribution Line [MDL] or account) level. Note: FIMA tracks at the document level versus CAMS that tracks at the item/MDL level).

In addition to the information that is recorded in the TRIAL table, the transaction tables will also include data obtained from the source tables that generated the TRIAL entry. The NOAA Data Mart will capture as much relevant data as possible that is static at the time the transaction was generated.

The transactions will record one or more of the following document numbers by name. The term in parentheses indicates the field prefix in the table:

- Commitment Number (COMMIT)
- Purchase Order Number (PO)
- Estimated Accrual Number (EA)
- Receiving Ticket Number (RT)
- Vendor Invoice Number (INV)
- Disbursement Number (DISB)
- Invoice Correction Number (INVCOR)
- Void Number (VOID)
- General Journal Number (GJ)
- Source Reference *(PRE)* This field is stored as REFERENCE_NUMBER and will appear with a prefix as reflected in many of the other document types in this list (e.g., PO_REFERENCE_NO, INV_REFERENCE_NO, etc.).
- Feeder System Number (PO_FEEDER_SYS_NO). Only entered on Purchase Orders. Will be carried forward with other Purchase Order related fields.

If the document is a Purchase Order, it will have a Purchase Order number. If the document is a Purchase Order that was based on a commitment, it will have both the Purchase Order number as well as the Commitment number. As mentioned previously, the transactions will be created with as much information as is available at the time the transaction is generated and will not be modified. Therefore, the Purchase Order will have the Commitment number, but the Commitment will not have the Purchase Order number (it is not known at the time the commitment is created).

The detailed transactions will capture all transaction numbers relevant at the time the transaction was generated providing a variety of ways in which to query the data.

3.2.3 NOAA Data Mart Support Table Definitions

There are a number of supporting tables that will be populated either through the batch jobs or via SQL commands by the system administrator (e.g., for parameter tables). These tables are defined in the following subsections.

3.2.3.1 NDW_ACCOUNT_PERIOD_STATUS

Description: This table will provide information on the status of the General Ledger Accounting Period based on the General Ledger beginning and ending dates. This status (O – Open, P – Preliminary Close, or C – Closed) will help users better interpret the data being reviewed. For example, unless the period is closed, not all batch processes may have been run and additional transactions may be processed that change the amounts.

Update Method: NOAA Data Warehouse Routine

3.2.3.2 NDW_ACCS_ID_CONTROL

Description: This table stores the unique NDW_ACCS_ID for each ACCS combination (based on BUREAU_CODE, FUND_CODE, ORG1-7_CODE, PROGRAM1-4_CODE, PROJECT_CODE, TASK_CODE, and OBJECT1-4_CODE). The NDW_ACCS_ID is then associated with the TRIAL_ID of the TRIAL record to which is applies. Once the NDW_ACCS_ID is determined for the TRIAL record, comparison of the NDW_ACCS_ID to summary tables is more efficient.

Update Method: NOAA Data Warehouse NDW902_REFRESH.SQL Routine

3.2.3.3 NDW_ACCS_ID_MAP

Description: One to many relationship between the NDW_ACCS_ID_CONTROL table and NDW_ACCS_ID_MAP table. Associates the NDW_ACCS_ID to the TRIAL_ID of the TRIAL record to which is applies.

Update Method: NOAA Data Warehouse NDW902_REFRESH.SQL Routine

3.2.3.4 NDW_DEFAULTS

Description: Stores default parameters to be used by the NOAA Data Mart processes. These defaults apply to the entire data mart instance (versus any single bureau, ASC, Line Office, or FMC. Parameters are defined through the ITEM_NAME with the value stored in the VALUE field. The purpose of this table is to eliminate any hard coding of values that may change over time within the code.

Update Method: SQL script to initially populate as well as SQL commands executed by the NOAA Data Mart administrator. Eventually, an on-line screen may be developed for data entry.

3.2.3.5 NDW_EXTRACT_ID_TABLE

Description: Assigns a unique identifier to each Line/Field Office extract permitted to record extracts against the NOAA Data Mart tables. Identifies the Bureau, Line Office, and FMC performing the extract; the interfacing system name, the point-of-contact employee number and name, and any notes concerning the extract.

Update Method: SQL script to initially populate as well as SQL commands executed by the NOAA Data Mart administrator. Eventually, an on-line screen may be developed for data entry.

3.2.3.6 NDW_FIN_CAT_DEF_CONTROL

Description: Defines the Financial Category to be used to summarize TRIAL data. Provides a start and end date during which this category was used and its current active status. Any changes to a definition must be done by Fiscal Year.

Update Method: SQL script to initially populate as well as SQL commands executed by the NOAA Data Mart administrator. Eventually, an on-line screen may be developed for data entry.

3.2.3.7 NDW_FIN_CAT_DEF_DETAIL

Description: Defines the Standard General Ledger account/sub-account numbers that will be summarized within the associated Financial Category. Provides a start and end date during which this account number was included and its current active status. Any changes to a definition must be done by Fiscal Year.

Update Method: SQL script to initially populate as well as SQL commands executed by the NOAA Data Mart administrator. Eventually, an on-line screen will be developed for data entry.

3.2.3.8 NDW_LO_EXTRACT_LOG

Description: Provides a log of the Line/Field Office routine that performed the extract, the date and time that the routine started and ended, the beginning and ending trans numbers processed, beginning and ending modification dates processed, the table name against which the extract was performed, and the number of records and amount processed and extracted.

This log will be populated by the Line Office extract routines through logic they develop.

Update Method: Line/Field Office Extract Routines

3.2.3.9 NDW_MAXSEQNOS

Description: Controls the next sequential number to be used for table transaction numbers / control fields.

Update Method: SQL script to initially populate/update for new entries. NOAA Data Warehouse Refresh Routines as numbers are assigned.

3.2.3.10 NDW_PROCESS_LOG

Description: A detailed log of the activities performed by a NOAA Data Mart batch routine. This table is intended to assist in tracking the status of a routine and for debugging in the event of an abnormal termination. Entries are generally included in the .log reports generated by the batch routine.

Update Method: NOAA Data Warehouse Batch Routines

3.2.3.11 NDW_REFRESH_PARAMS

Description: Stores parameters to be used by the NOAA Data Mart refresh routines that populate the data mart tables. These routines will generally maintain the last transaction number or modification date processed for a particular table.

Update Method: NOAA Data Warehouse Refresh Routines

3.2.3.12 NDW_REFRESH_RUN_CONTROL

Description: Stores information concerning the data to be included in the refresh run that is initiated. Also tracks the status of the various refresh routines to ensure successful completion and restart if necessary.

Update Method: Entry is initially created through NDW901_INITIATE.SQL and then updated by the other refresh routines to reflect status.

3.2.3.13 NDW_RETURN_CODE_CONTROL

Description: Contains a unique message indicator, a short description for display on reports/forms, and a long description for troubleshooting. This table ensures consistency of message text for the same event.

Update Method: SQL script to initially populate as well as SQL commands executed by the NOAA Data Mart administrator. Eventually, an on-line screen may be developed for data entry.

3.2.4 CAMS Data Warehouse Reference Table Definitions

The CAMS Data Warehouse refresh routine takes snapshots of a number of CFS reference tables. These reference tables will be available for user access for query only. Note that discussions with Finance still need to take place to determine what data elements should be made available for some of these tables (e.g., vendor and customer tables). The table formats are included in Appendix D – Reference Table Record Layouts.

- **CUSTOMER** – Information concerning customers for accounts receivable transactions.

- **CUSTOMER_CONTACT** – One-to-many child relationship to CUSTOMER_CONTROL. Identifies customer points-of-contact.

- **DW_BUREAU_DIM** – Identifies the bureau code, name, and active status for each bureau.

- **DW_EMPLOYEE_DIM** – Provides some basic information concerning an employee (i.e., employee number, name, and e-mail address).

- **FUND** – Identifies the fund code, title, and active status for each fund.

- **OBJECT1** – Identifies the object1 code, description, and active status for the first level of the object code.

- **OBJECT2** – Identifies the object1-2 codes, description, and active status for the second level of the object code.

- **OBJECT3** – Identifies the object1-3 codes, description, and active status for the third level of the object code.

- **OBJECT4** – Identifies the object1-4 codes, description, and active status for the fourth level of the object code.

- **ORG1** – Identifies the org1 code, description, and active status for the first level of the organization code.

- **ORG2** – Identifies the org1-2 codes, description, and active status for the second level of the organization code.

- **ORG3** – Identifies the org1-3 codes, description, and active status for the third level of the organization code.

- **ORG4** – Identifies the org1-4 codes, description, and active status for the fourth level of the organization code.

- **ORG5** – Identifies the org1-5 codes, description, and active status for the fifth level of the organization code.

- **ORG6** – Identifies the org1-6 codes, description, and active status for the sixth level of the organization code.

- **ORG7** – Identifies the org1-7 codes, description, and active status for the seventh level of the organization code.

- **DW_PAYMENT_OFFICE_DIM** – Identifies the payment office code and the name.

- **PROGRAM1** – Identifies the program1 code, description, and active status for the first level of the program code.

- **PROGRAM2** – Identifies the program1-2 codes, description, and active status for the second level of the program code.

- **PROGRAM3** – Identifies the program1-3 codes, description, and active status for the third level of the program code.

- **PROGRAM4** – Identifies the program1-4 code, description, and active status for the fourth level of the program code.

- **PROJECT** – Identifies the project code, fund and program1-4 codes associated it, description, and active status.

- **TASK** – Identifies the project code, task code, description, and active status.

- **VENDOR_CONTROL** – Provides information about vendors.

- **VENDOR_DETAIL** – One-to-many child relationship to VENDOR_CONTROL. Provides address information about a vendor. One vendor may have multiple addresses (e.g., one address for purchasing, a different address for payments).

3.3 NOAA Data Mart Population

Immediately after the CAMS Data Warehouse Update is complete, the NOAA Data Mart Refresh process will be started. This process will execute a series of routines to obtain new records from the CFS TRIAL table, newly approved records from BUDGET_CONTROL and BUDGET_DETAIL, newly approved labor transactions from GJ_CONTROL, GJ_DETAIL, and GJ_EMPLOYEE, and delta snapshots of CFS source tables to update the NOAA Data Mart tables.

These routines will be launched and controlled through a UNIX shell script to be run through a cron tab.

The following sections summarize the method in which the various NOAA Data Mart tables will be populated.

3.3.1 Refresh Initiation

The NOAA Data Mart refresh initiation process (NDW901_INITIATE.SQL) will allow either a new refresh to be started or a stopped refresh to be restarted. This routine accepts two parameters:

Parameter 1: Indicates whether the routine is a new refresh (NEW) or a restart of a stopped refresh (RESTART).

Parameter 2: Indicates the Refresh ID Number to be restarted.

This routine performs the following major functions:

- Ensures that the operator has not requested that all NDW batch processes be stopped. If this is the case, a NDW refresh cannot be run.

- If the request is to restart a previous refresh run, ensure that the run exists and that it was stopped before all transactions could be processed. Update the Refresh Restart counter by 1.

- If the request is to start a new refresh, ensure there are no active NDW refresh jobs running (only one NDW refresh process should be running at a time).

- Determine the maximum TRIAL_ID's for all records and for those in specified groups. These TRIAL_ID's will be used to determine if a job has run through completion or not.

- Determine the maximum MODIFICATION_DATE and TRANS_NO combination for newly approved BUDGET_CONTROL records. These key fields will be used to determine if the BOP refresh jobs have run through completion or not.

- Determine the maximum MODIFICATION_DATE and TRANS_NO combination for newly approved GJ_CONTROL records. These key fields will be used to determine if the labor refresh jobs have run through completion or not.

- Will compare the maximum fields to the previous refresh to determine if there is a difference. If ALL of the fields are the same as those from the previous refresh, a message will be written to the NDW_PROCESS_LOG and the refresh will be stopped.

- Create an entry in the NDW_REFRESH_RUN_CONTROL table that assigns a unique Refresh Run ID to the job. This entry is used to track the status of the refresh run.

Detailed specifications for this routine are provided in Section 4.1.1, Initiate NOAA Data Warehouse Refresh Process.

3.3.2 Assigning Unique ACCS ID'S

In order to improve performance for the batch updating of data mart tables from TRIAL, the NOAA Data Mart will assign a unique ACCS ID code to each unique ACCS combination found within TRIAL. Once assigned, the comparison between ACCS codes between the detail and summary tables will be performed by the NDW_ACCS_ID (defined as a NUMBER field) versus the combination of ACCS elements (47 positions).

The NDW902_REFRESH.SQL routine performs the following major functions:

- For each new TRIAL record within the NOAA Data Mart, compare the relevant ACCS elements to the elements in the NDW_ACCS_ID_CONTROL table.

- For each new combination, create an entry in the NDW_ACCS_ID_CONTROL table and a subordinate entry in the NDW_ACCS_ID_MAP table reflecting the TRIAL_ID.

- If the combination already exists, create a subordinate entry in the NDW_ACCS_ID_MAP table reflecting the TRIAL_ID.

Detailed specifications for this routine are provided in Section 4.1.2, Assign ACCS ID's .

3.3.3 Refreshing Tables

The NOAA Data Mart is comprised of several refresh routines to populate the tables that may be run in parallel. These processes follow a standard set of guidelines that will not be repeated in the subsections that follow:

- Obtain the Refresh ID Number from the NDW_REFRESH_RUN_CONTROL table where the ACTIVE_STATUS field is set to “Y”. This number is used to uniquely identify the refresh run. The table entry contains control total and status information concerning the refresh process for this run. Log entries are made and the process is terminated if either no entry or more than one entry is found.

- Increments the Pass Number for the specific refresh routine by 1. Refresh routines can have one of two or both of the following parameters in the NDW_DEFAULTS table associated with it. The first, *ROUTINE*_RECORDS_TO_ PROCESS, identifies how many records to process at a time. This allows the routine to segment larger volumes of data into smaller processing increments if necessary for better performance. If there are more records to be refreshed than the number of records this routine will process at a time, additional passes will be performed. The second parameter, *ROUTINE*_ RECORDS_TO_COMMIT indicates how often to commit. This helps control the database rollback segment from getting too large and commits the process log entries to aid in monitoring job progress and debugging.

- Checks to see if the CFS snapshots are complete. If the snapshots of CFS production tables has not completed, the routine will be terminated.

- At the beginning of the job and at each commitment point, checks to see if the operator has requested that NDW batch processes should be stopped (END_NDW_BATCH_ PROCESSES_FLAG in the NDW_DEFAULTS table). If this flag is set to “Y”, log and control records are updated and the process is terminated.

- Obtains the last control fields that were processed by this routine. The control field used may vary from process to process. For example, the routine to refresh the NDW_GL_ACCT_SUMMARY table uses the TRIAL_ID field as it’s indicator of the last record processed. However, the routine to refresh the NDW_BOP_SUMMARY uses a combination of the BUDGET_CONTROL MODIFICATION_DATE and TRANS_NO. The values should be the same between processes, but may be different if the operator stopped the process (versus the jobs running through completion). These fields are then updated at the completion of the process.

- Whenever ANY transaction record is created, the NDW_CREATION_DATE, NDW_CREATION_USER_NAME, and NDW_CREATION_DEVICE fields are updated.

- Whenever ANY transaction or support table record is updated, the NDW_LAST_MOD_DATE, NDW_LAST_MOD_USER_NAME, and NDW_LAST_MOD_DEVICE fields are updated.

- The number of records and a control total are tracked for each refresh routine and these figures are reflected in the NDW_PROCESS_LOG at each commit point and at the end of the process.

- Log records are written at the start of the job, whenever an exception condition is encountered, and at the end of the job. These records are retained within the NDW_PROCESS_LOG table and are also written as a formatted report which is stored on the server using a standard naming convention.

- The NDW_REFRESH_RUN_CONTROL record for the refresh run is updated at the end of the routine to indicate the last pass number and the maximum value processed. This entry allows the operator to easily review the status of each of the processes for a specific refresh run.

3.3.3.1 Refresh NDW_GL_ACCT_SUMMARY

The NDW_GL_ACCT_SUMMARY table summarizes TRIAL data by standard general ledger account/sub-account number, month and fiscal year (based on general ledger accounting period), and ACCS. This groups the data at the lowest standard general ledger level for all account numbers that can be generated through CFS – not just those required by the Line/Field Office commitment tracking/MIS interfaces.

The NDW001_REFRESH.SQL routine performs the following major functions:

- Read a specified number of TRIAL records and group them by ACCOUNT_NO, SUB_ACCOUNT_NO, GL_END_DATE, FISCAL_YEAR, FUND_CODE_FISCAL_YEAR, NDW_ACCS_ID (by linking TRIAL_ID to

NDW_ACCS_ID_MAP table), and BALANCE_FLAG (by linking ACCOUNT_NO and SUB_ACCOUNT_NO to CHART table).

- Match the NDW_ACCS_ID, fund code fiscal year, and GL_END_DATE to the elements in the NDW_GL_ACCT_SUMMARY table.
- If a match is found, add the TRIAL credit amount, debit amount, net amount, and labor hours (if applicable) to the NDW_GL_ACCT_SUMMARY values.
- If match is not found, create an entry with the credit amount, debit amount, net amount, and labor hours (if applicable).

Detailed specifications for this routine are provided in Section 4.2, NOAA Data Mart Summary Table Refresh (NDW001_REFRESH.SQL).

3.3.3.2 Refresh NDW_FIN_CAT_SUMMARY

The NDW_FIN_CAT_SUMMARY table summarizes TRIAL data at one level higher than the NDW_GL_ACCT_SUMMARY. In this table, one or more standard general ledger accounts/sub-accounts are combined into one or more NOAA-defined categories. Categories are defined or can be changed at the beginning of a fiscal year. The definition of financial categories as defined by NOAA for this table is shown in Figure 7.

Financial Category	Account Number	Description
COMMIT – Commitments		
COMMIT	4700	COMMITMENTS
OBLIG - Obligations		
OBLIG	4801	UNDELIVERED ORDERS - OBLIGATIONS, UNPAID
OBLIG	4802	UNDELIVERED ORDERS - OBLIGATIONS, PREPAID/ADVANCED
OBLIG	4881	UPWARD ADJ OF PY UDO-OBLIGATIONS, UNPAID
OBLIG	4882	UPWARD ADJ OF PY UDO-OBLIGATIONS, PREPAID/ADVANCED
OBLIG	4901	DELIVERED ORDERS - OBLIGATIONS, UNPAID
OBLIG	4902	DELIVERED ORDERS - OBLIGATIONS, PAID
OBLIG	4981	UPWARD ADJ OF PY DELIVERED ORDERS-OBLIGATIONS, UNPD
OBLIG	4982	UPWARD ADJUSTMENT OF PY DEL ORDERS-OBLIGATIONS, PD
UDO – Undelivered Orders		
UDO	4801	UNDELIVERED ORDERS - OBLIGATIONS, UNPAID
UDO	4802	UNDELIVERED ORDERS - OBLIGATIONS, PREPAID/ADVANCED

Financial Category	Account Number	Description
UDO	4881	UPWARD ADJ OF PY UDO-OBLIGATIONS, UNPAID
UDO	4882	UPWARD ADJ OF PY UDO-OBLIGATIONS, PREPAID/ADVANCED
UDO_DWADJ – Undelivered Orders – Downward Adjustments		
UDO_DWADJ	4871	DOWNWARD ADJ OF PY UNPAID UDO-OBLIGATIONS, RECOVERY
UDO_DWADJ	4872	DOWNWARD ADJ OF PY PREPAID/ADV UDO-OBL, REFUNDS COL
UEXP – Unpaid Expenses		
UEXP	4901	DELIVERED ORDERS - OBLIGATIONS, UNPAID
UEXP	4981	UPWARD ADJ OF PY DELIVERED ORDERS-OBLIGATIONS, UNPD
PEXP – Paid Expenses		
PEXP	4902	DELIVERED ORDERS - OBLIGATIONS, PAID
PEXP	4982	UPWARD ADJUSTMENT OF PY DEL ORDERS-OBLIGATIONS, PD
UEXP_DWADJ – Unpaid Expenses – Downward Adjustments		
UEXP_DWADJ	4971	DOWNWARD ADJ OF PY UNPD DELIVERED ORDERS-OBL, RECOV
PEXP_DWADJ – Paid Expenses – Downward Adjustments		
PEXP_DWADJ	4972	DOWNWARD ADJ OF PY PAID DEL ORDERS-OBL, REFUNDS COL

Figure 7. NOAA Data Mart Financial Category Definitions

The NDW002_REFRESH.SQL routine performs the following major functions:

- For each new TRIAL record within the NOAA Data Mart, determine the financial category(s) under which the record applies. Note that one transaction can be included under more than one financial category. For example, NOAA's definition of an obligation is equal to the undelivered orders plus unpaid expenses plus paid expenses. Therefore, the amount for an undelivered order transaction would be included in both the OBLIG and UDO financial categories.

- Match the TRIAL_ID to the TRIAL_ID in the NDW_ACCS_ID_MAP table to obtain the NDW_ACCS_ID for the TRIAL record.

- Match the NDW_ACCS_ID, fund code fiscal year, and GL_END_DATE to the elements in the NDW_FIN_CAT_SUMMARY table.

- If a match is found, compute the net amount and add the net amount and labor hours (if applicable) to the NDW_FIN_CAT_SUMMARY values fields for the financial category(s) to which the transaction belongs. Note that the NDW_FIN_CAT_SUMMARY shows the values for all defined categories within the row. Therefore, only a single row needs to be read to obtain all financial category amounts for a unique ACCS for the month.

- If match is not found, compute the net amount and create an entry setting the net amount and labor hours (if applicable) to the NDW_FIN_CAT_SUMMARY value fields for the financial category(s) to which the transaction belongs.

Detailed specifications for this routine are provided in Section 4.2.2, Refresh NDW_FIN_CAT_SUMMARY Table.

3.3.3.3 Refresh NDW_BOP_SUMMARY

Budget Operating Plans are created within CFS through the FM066 screen, Budget Operating Plan Transaction Screen. The NDW_BOP_SUMMARY table summarizes BOP data at the Fiscal Year, budget month, ACCS level.

The NDW003_REFRESH.SQL routine performs the following major functions:

- For each newly approved BUDGET_CONTROL and BUDGET_DETAIL record within the NOAA Data Mart, match the ACCS elements, fiscal year, and budget month to the elements in the NDW_BOP_SUMMARY table.

- If a match is found, add the BUDGET_DETAIL amount and STAT_UNIT_QTY (FTE amount) to the NDW_BOP_SUMMARY values.

- If match is not found, create an entry with the amount and STAT_UNIT_QTY (FTE amount).

Detailed specifications for this routine are provided in Section 4.2.3, Refresh NDW_BOP_SUMMARY Table.

3.3.3.4 Refresh NDW_COMMIT_TRANS

The NDW010_REFRESH.SQL routine will update the NDW_COMMIT_ TRANS table with new TRIAL records.

Commitments are identified in TRIAL as having an ACCOUNT_NO = 4700. As commitments are currently not being processed within CFS, they are considered a priority 3 item and will be addressed at a later point.

3.3.3.5 Refresh NDW_AP_TRANS

The NDW011_REFRESH.SQL routine will update the NDW_AP_TRANS table with new TRIAL obligation and expense records.

Obligations are identified in TRIAL as having an ACCOUNT_NO between 4800 and 4899. Expenses are identified in TRIAL as having an ACCOUNT_NO between 4900 and 4999. The definitions for the obligation and expense standard general ledger account numbers are provided in Figure 8. The NDW_AP_TRANS refresh routine will process all 4800 through 4999 standard general ledger account numbers.

Account Number	Description
Obligation Accounts	
4800	UNDELIVERED ORDERS
4801	UNDELIVERED ORDERS - OBLIGATIONS, UNPAID
4802	UNDELIVERED ORDERS - OBLIGATIONS, PREPAID/ADVANCED
4831	UNDELIVERED ORDERS, TRANSFERRED-UNPAID – IN
4831	UNDELIVERED ORDERS, TRANSFERRED-UNPAID - OUT
4832	UNDELIVERED ORDERS, TRANSFERRED-PAID – IN
4832	UNDELIVERED ORDERS, TRANSFERRED-PAID – OUT
4870	DOWNWARD ADJ OF PY UDO
4871	DOWNWARD ADJ OF PY UNPAID UDO-OBLIGATIONS, RECOVERY
4872	DOWNWARD ADJ OF PY PREPAID/ADV UDO-OBL, REFUNDS COL
4880	UPWARD ADJ OF PY UDO
4881	UPWARD ADJ OF PY UDO-OBLIGATIONS, UNPAID
4882	UPWARD ADJ OF PY UDO-OBLIGATIONS, PREPAID/ADVANCED
Expense Accounts	
4900	EXPENDED AUTHORITY
4901	DELIVERED ORDERS - OBLIGATIONS, UNPAID
4902	DELIVERED ORDERS - OBLIGATIONS, PAID
4931	DELIVERED ORDERS -OBLIGATIONS TRANSFERRED, UNPD-IN
4931	DELIVERED ORDERS -OBLIGATIONS TRANSFERRED, UNPD-OUT
4971	DOWNWARD ADJ OF PY UNPD DELIVERED ORDERS-OBL, RECOV
4972	DOWNWARD ADJ OF PY PAID DEL ORDERS-OBL, REFUNDS COL
4979	DOWNWARD ADJ OF PY EXP AUTH - OTHER
4980	UPWARD ADJUSTMENT OF PY EXPENDED AUTHORITY
4981	UPWARD ADJ OF PY DELIVERED ORDERS-OBLIGATIONS, UNPD
4982	UPWARD ADJUSTMENT OF PY DEL ORDERS-OBLIGATIONS, PD

Figure 8. Obligation and Expense Account Numbers in Trial

The NDW011_REFRESH.SQL routine performs the following major functions:

- Select new TRIAL records within the NOAA Data Mart with an ACCOUNT_NO between 4800 and 4999.

- Determine the type of transaction based on the ACCOUNT_NO, SUBSYSTEM_CODE, and TRANS_SOURCE.

- Based on the type of transaction, refer back to the source tables to obtain the information required in the NDW_AP_TRANS table.

- Determine if the transaction is a reversing entry for that account type and tag it as such. The transaction tables will store both the transaction that created the charge (e.g., a purchase order creates an unpaid, undelivered order transaction) as well as a transaction that reverses (or liquidates) the charge. For example, when a receiving ticket is received, the unpaid, undelivered order charge is liquidated and the charge now becomes an unpaid, delivered order.

- Determine if the transaction is a prior year transaction and tag it as such.

- Determine if the transaction is a labor record and tag it as such.

- Populate the fields within the NDW_AP_TRANS table and stores the record.

Detailed specifications for this routine are provided in Section 4.3.2, Refresh Accounts Payable Transaction Table From Trial.

3.3.3.6 Refresh NDW_BOP_DETAIL

The NDW012_REFRESH.SQL routine will update the NDW_BOP_DETAIL table with newly approved Budget Operating Plan data.

Newly approved BOPs will be selected as having an APPROVED_FLAG = "Y" and a MODIFICATION_DATE (includes time) and TRANS_NO combination greater than the last modification date/time and TRANS_NO combination previously processed.

The NDW012_REFRESH.SQL routine performs the following major functions:

- Select newly approved BUDGET_CONTROL and BUDGET_DETAIL records within the NOAA Data Mart.

- For each new BUDGET_DETAIL record, create an entry in the NDW_BOP_DETAIL table.

Detailed specifications for this routine are provided in Section 4.3.3, Refresh Transaction Tables From Budget Tables.

3.3.3.7 Refresh NDW_LABOR_DETAIL

The NDW013_REFRESH.SQL routine will update the NDW_LABOR_DETAIL table with new GJ_CONTROL, GJ_DETAIL, and GJ_EMPLOYEE labor data resulting from the NFC interface (NFC002), labor detail adjustments made through the CFS NFC005 screen, and month end transaction defaults generated through the NFC004 routine.

The NDW013_REFRESH.SQL routine performs the following major functions:

- Select newly approved GJ_CONTROL, GJ_DETAIL, and GJ_EMPLOYEE records within the NOAA Data Mart.

- For each new GJ_EMPLOYEE record, create an entry in the NDW_LABOR_DETAIL table.

Detailed specifications for this routine are provided in Section 4.3.4, Refresh Detailed Labor Data (NDW013_REFRESH.SQL).

3.3.3.8 Refresh NDW_RESERV_TRANS

The logic for populating the NDW_RESERV_TRANS table will be provided as part of priority 2 efforts (not in the initial phase).

3.4 Data Integrity

A number of mechanisms will be implemented as part of the NOAA Data Mart to ensure data integrity. These include:

- Storing relevant data within the NDW_REFRESH_RUN_CONTROL table that identifies the target numbers to be processed and the status of the refresh routines in meeting those targets.

- Frequent entries within the NDW_PROCESS_LOG table to record the beginning and ending status of batch processes, and interim status messages for performance monitoring and debugging.

- A refresh status report that will compare control totals between the TRIAL and relevant CFS source tables to the NOAA Data Mart tables to ensure they balance.

3.5 Line Office Extract Processing

Each Line/Field Office will be responsible for developing the extract routines necessary to obtain the data for their commitment tracking systems.

Several fields have been established within each table from which Line/Field Offices may extract data. These fields are provided so that the Line/Field Office may indicate that a row was extracted (and for some tables may not want to be extracted again) and track how many times a row is extracted.

For the transaction tables, once a row is populated, it should not be changed. Therefore, the extracting routine need only extract any particular row once.

For the summary tables, any open accounting period may have data updated until that accounting period is closed. In these cases, the Line/Field Office may wish to re-extract either all records on a daily basis or only those that have been changed. Note that the amounts on summary table records are changed to reflect the new transactions processed – the delta is not stored. Records may be identified as being changed after a Line/Field Office extract by the NDW_LAST_MOD_DATE being greater than the Line/Field Office LAST_EXTRACT_DATE.

There is one group of extract fields provided for each Line Office plus an additional ten groups for other commitment tracking/MIS systems requiring separate extracts. The Line Office extract fields are all prefixed by the Line Office acronym (e.g., "OFA_", "OMAO_", "NOS_", etc.). The additional groups have a sequential number prefix beginning with "01_" through "10_".

The extract fields that are provided are shown in Figure 9 (note that the "OFA_" prefix is used as an example):

Extract Field Name	Data Type	Definition
OFA_ORIGINAL_EXTRACT_DATE	DATE	The date that this record was originally extracted by this Line/Field Office. Will be populated with the system date.

Extract Field Name	Data Type	Definition
OFA_ORIGINAL_USER_ NAME	VARCHAR2(30)	Name of user who executed the routine that originally extracted data from this record for this Line/Field Office. Will be populated with the user's name derived from the Oracle user account of the user running the extract.
OFA_ORIGINAL_DEVICE_ NAME	VARCHAR2(30)	Name of device used when the record was originally extracted for this Line/Field Office. Will be populated with the word "BATCH" followed by the name of the routine that extracted the record.
OFA_LAST_EXTRACT_DATE	DATE	The last date that data was extracted from this record for this Line/Field Office. Will be populated with the system date.
OFA_LAST_USER_NAME	VARCHAR2(30)	Name of user who executed the routine that last extracted data from this record for this Line/Field Office. Will be populated with the user's name derived from the Oracle user account of the user running the extract.
OFA_LAST_DEVICE_NAME	VARCHAR2(30)	Name of device used when the record was last extracted for this Line/Field Office. Will be populated with the word "BATCH" followed by the name of the routine that extracted the record.
OFA_EXTRACT_ITERATION	NUMBER(2)	The number of times the data from this record was extracted for this Line/Field Office. Will be incremented by 1 each time the record is extracted by this Line/Field Office.

Figure 9. Line/Field Office Extract Fields

A database package (NDW_RECORD_LO_EXTRACT) will be developed as part of the NOAA Data Mart for Line/Field Office extract use. This package will perform the following major functions:

- Calling routine will pass the Line/Field Office extract prefix (e.g., "OFA", "OMAO", "01", "02", etc.), the table name and

NDW_TRANS_NO of the record being extracted, and the name of the routine performing the extract.

- If the table or row cannot be found, a message will be returned to the calling routine.

- The database package will determine the User Name.

- Package will check if the *pre*_ORIGINAL_EXTRACT_DATE is null for the row ID of the table requested. If so, it will update the ORIGINAL extract fields as shown in the table.

- Will populate the *pre*_LAST_MOD extract fields as shown in the table for all extracts (for the original extract, the ORIGINAL and LAST_MOD values will be the same).

Detailed specifications for this routine are provided in Section 4.4.1, NDW_RECORD_LO_EXTRACT.

3.6 Line Office On-Line Access

In addition to the designated users having access to the NOAA Data Mart tables for extracting to the Line/Field Office commitment tracking/MIS systems, much of this data will also be available to managers and finance users. On-line access for users will be available as customized reports and queries (to be developed as a separate effort) through the Navigator Menu.

Oracle Discoverer will also be implemented as a web-based capability for users accessing data through defined business areas and shared worksheets (predefined queries) or for producing their own ad-hoc queries. Note that the implementation of Oracle Discoverer, definition of business areas, and development of on-line queries and reports through Discoverer are not being performed within this project. They are being performed through other NOAA efforts.

The specific tables and data elements users will be able to access on-line will be enforced through database roles as described in Section 3.8, Data Security.

3.7 Refresh Recovery

The NOAA Data Warehouse project will be defining the requirements, standards, and logic for ensuring that data populated through refresh routines can be backed out and repopulated if necessary. Once these standards are defined, the approach will be incorporated into the NOAA Data Mart.

3.8 Data Security

The NOAA Data Warehouse (NDW) is implemented under the Oracle Relational Database Management System (RDBMS) Version 8.1.6.x running on the Cumulus Server located at the ITC in Largo, Maryland. Access to any CAMS database and application software, requires that a user is authorized and has a valid user ID and password. All CAMS data and applications enforce controls, audit-ability, and access restrictions based on the NOAA CAMS approved security plan. CAMS applications restrict access based on functional and database-enforced roles, database views, segregation of duties, and presentation of menu options/applications based on preauthorized access approval. Privacy Act, procurement sensitive, and non-classified business sensitive data are all protected within approved NOAA guidelines. User access to data is limited based on the appropriate controls enforced at the database, application, table, and data content level (e.g., via roles and views).

The NDW will enforce at least three (3) user access levels to data within the data warehouse. All controls specified above are enforced with the additional clarification on access to the data, including:

1. Extract users who will have access to all data related to their organization for the purposes of populating official organizational commitment tracking/MIS systems. Extract users are generally privileged, batch oriented programs that extract data from the NDW for purposes of populating a different system.

2. Full access (all data content for a particular organization) to all authorized data and associated NDW applications. There may be multiple roles or database views depending on the data/tables that need to be accessed. The use of database roles and views will enable access to all columns within a row.

3. End-user access is restricted to a subset of data and tables that the user is authorized to view. Database roles and views limit or restrict access to columns in rows where privacy act and/or other sensitive information may reside.

Data access via Open Database Connectivity (ODBC) shall not be supported.

3.8.1 Database Roles

Access to NOAA Data Warehouse tables, rows within tables, and columns within a row for formal extract, update, and query will be enforced through database roles and views assigned to users. There are at least two (2) template roles that will exist:

- **NDW_USER** – Assigned to all users who will have access to the NOAA Data Mart. A view(s) that allows access to unrestricted data is associated with this role.

- **NDW_PRIV_ACT_USER** – Assigned to those users who will have access to records within the NOAA Data Mart with a PRIVACY_ACT_APPLIES = "Y". There may be multiple views associated and possibly multiple roles based on this template.

The templates will be used to construct roles and views that allow for access to be controlled based on the type of data presented in the data warehouse. Transaction data segregated by generic, labor, privacy act, procurement sensitive, etc. will need views and associated roles to limit which information is generally available via extract and on-line query access.

3.8.2 Privacy Act Data

Each table has a column titled "PRIVACY_ACT_APPLIES". Based on NOAA-defined business rules, records containing data to which the Privacy Act applies will have this field populated with a "Y".

A separate database role will be established and granted to users who should have access to data controlled by the Privacy Act. Users without this role will not be able to access those records.

4. Detailed Design

4.1 NOAA Data Mart Refresh

In addition to the refresh routines that populate the various NOAA Data Mart tables, the system includes several other routines that manage the refresh process and perform data integrity checks.

The NOAA Data Mart includes the refresh routines identified Figure 10 that are describe in detail in the sections that follow.

Routine Name	Routine Description
NDW901_INITIATE.SQL	Initiates the start of a NOAA Data Warehouse Refresh.
NDW902_REFRESH.SQL	Updates the NDW_ACCS_ID_CONTROL and NDW_ACCS_ID_MAP tables to reflect unique ACCS combinations.
NDW001_REFRESH.SQL	Populates the NDW_GL_ACCT_SUMMARY table.
NDW002_REFRESH.SQL	Populates the NDW_FIN_CAT_SUMMARY table.
NDW003_REFRESH.SQL	Populates the NDW_BOP_SUMMARY table.
NDW010_REFRESH.SQL	Populates the NDW_COMMIT_TRANS (priority 3) table.
NDW011_REFRESH.SQL	Populates the NDW_AP_TRANS table.
NDW012_REFRESH.SQL	Populates the NDW_BOP_DETAIL table.
NDW013_REFRESH.SQL	Populates the NDW_LABOR_DETAIL table.
NDW014_REFRESH.SQL	Populates the NDW_RESERV_TRANS table.
NDW050.SQL	Determines any change in a General Ledger accounting period's status and updates the MONTH_CLOSED_FLAG flag within the summary tables for that accounting period.

Figure 10. NOAA Data Mart Refresh Routines

4.1.1 Initiate NOAA Data Warehouse Refresh Process (NDW901_INITIATE.SQL)

The NDW901_INITIATE.SQL routine will launch the NOAA data warehouse refresh process and ensure that these processes are coordinated and run through completion. This process creates an entry in the NDW_REFRESH_RUN_CONTROL table, assigns a unique identifier to the refresh process, and determines the maximum TRIAL_ID numbers that should be processed for a routine to be considered complete. The subordinate refresh processes will then query this record and update it with their statuses as they complete.

Routine Name

NDW901_INITIATE.SQL

Input:

This routine will be invoked through a regularly scheduled cron job and uses the following data as input:

- Parameters
  - NEW or RESTART
  - REFRESH_ID_NO
- CAMS Data Warehouse Tables:
  - TRIAL (TR)
  - BUDGET_CONTROL (BC)
  - GJ_CONTROL (GC)
- NOAA Data Mart Tables:
  - NDW_MAXSEQNOS (NM)
  - NDW_DEFAULTS (ND)
  - NDW_REFRESH_RUN_CONTROL (NRRC)

Output:

This routine will produce the following return parameters, output files, database tables, and/or reports:

- NOAA Data Mart Tables:
  - NDW_REFRESH_RUN_CONTROL (NRRC)

    - NDW_MAXSEQNOS (NM)
    - NDW_PROCESS (NP)
- Reports:
    - NDW901_RE999_PASS000.log (if database is available).
    - NDW901_yyyymmdd.log (if database is not available).
    - Only keep 10 versions of NDW901 log files in the directory.

Processing Logic:

```
Begin Routine
Write date/time started to .log file.
Write header information to .log file.

Validate parameters.
If Parameter 1 does not = "NEW" or "RESTART"
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW901_INITIATE.SQL"
        DATE_TIME = system date/time
        STEP_DESCR = "NDW-000TBD:  NDW901_INITIATE Parameter 1 not equal to
        NEW or RESTART "
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

If Parameter 1 = "NEW" and Parameter 2 is specified
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW901_INITIATE.SQL"
        DATE_TIME = system date/time
        STEP_DESCR = "NDW-000TBD:  WARNING:  NDW901_INITIATE Parameter 1
        = NEW and Parameter 2 provided: " followed by the Parameter 2 value.
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Do not terminate processing.

Determine if a refresh is already running.  There should be only one refresh run active at
any time, although multiple processes may be run in parallel.
Read REFRESH_ID_NO
    From NDW_REFRESH_RUN_CONTROL (NRRC)
    Where ACTIVE_STATUS = "Y".

If an entry is found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW901_INITIATE.SQL"
        DATE_TIME = system date/time
        REFRESH_ID_NO = NRRC.REFRESH_ID_NO
        STEP_DESCR = "NDW-000TBD:  NDW refresh already in process with
        REFRESH_ID_NO noted."
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

- ***Determine if ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”***

```
Read VALUE(s)
    From NDW_DEFAULTS (ND)
    For ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”

If entry cannot be found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW901_INITIATE.SQL”
        REFRESH_ID = Parameter 2
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Entry not found in NDW_DEFAULTS for
        ITEM_NAME: ??”.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW901_INITIATE.SQL”
        REFRESH_ID = Parameter 2
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

- ***Determine if CAMS Data Warehouse snapshots are complete.***

If they are not complete, set the CFS_SNAPSHOTS_COMPLETE field in the NDW_REFRESH_RUN_CONTROL table to “N”.

RESTART REFRESH STOPPED BY OPERATOR

- ***Restart refresh that was previously stopped by the operator.***

```
If Parameter 1 = “RESTART”
```

  - ***Find refresh routine specified in parameter.***

```
    Read REFRESH_ID_NO, REFRESH_STOPPED_BY_OPERATOR,
    REFRESH_RESTART_NO
        From NDW_REFRESH_RUN_CONTROL (NRRC)
        Where NRRC.REFRESH_ID_NO = Parameter 2

    If an entry is not found
        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW901_INITIATE.SQL”
            DATE_TIME = system date/time
            REFRESH_ID_NO = Parameter 2
            STEP_DESCR = “NDW-000TBD:  NDW refresh job to restart not found.”
            All other fields remain null.
        Write NDW_PROCESS_LOG entry to .log file.
        Terminate processing.

    If NRRC.REFRESH_STOPPED_BY_OPERATOR = “N”
```

```
        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW901_INITIATE.SQL”
            DATE_TIME = system date/time
            REFRESH_ID_NO = Parameter 2
            STEP_DESCR = “NDW-000TBD:  NDW refresh job was not stopped by the
            operator.”
            All other fields remain null.
        Write NDW_PROCESS_LOG entry to .log file.
        Terminate processing.

    Update NDW_REFRESH_RUN_CONTROL
    Update NDW_REFRESH_RUN_CONTROL
        For record with REFRESH_ID_NO = Parameter 2
        Set REFRESH_STOPPED_BY_OPERATOR to “N”
        Set ACTIVE_STATUS = “Y”
        Set REFRESH_RESTART_NO to NRRC.REFRESH_RESTART_NO + 1

    If record cannot be updated:
        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW901_INITIATE.SQL”
            REFRESH_ID = Parameter 2
            DATE_TIME = system date/time
            STEP_DESCR = “NDW-000TBD:  Could not update
            NDW_REFRESH_RUN_CONTROL table.”
            All other fields remain null.
        Write NDW_PROCESS_LOG entry to .log file.
        Terminate processing.

    Perform Close Routine.
Else

START NEW REFRESH

Determine next REFRESH_RUN_ID.  There should be only one refresh run active at any
time, although multiple processes may be run in parallel.
Obtain next sequential number from NDW_MAXSEQNOS and update table
    Where TABLE_NAME = “NDW_REFRESH_RUN_CONTROL”
If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW901_INITIATE.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No entry found in NDW_MAXSEQNOS for
        TABLE_NAME = NDW_REFRESH_RUN_CONTROL.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

Determine maximum TRIAL_ID values.
Read from TRIAL to determine:
    Maximum TRIAL_ID (for all account numbers)
    Maximum TRIAL_ID for records with ACCOUNT_NO = 4700
    Maximum TRIAL_ID for records with ACCOUNT_NO between 4800 and 4899
    Maximum TRIAL_ID for records with ACCOUNT_NO between 4900 and 4999

Determine maximum BUDGET_CONTROL key field values.
```

```
Read from BUDGET_CONTROL to determine:
    Where APPROVED_FLAG = “Y”
    Maximum MODIFICATION_DATE || TRANS_NO
```

Determine maximum GJ_CONTROL key field values (for labor detail).

```
Read from GJ_CONTROL to determine:
    Where MANAGER_FLAG = “Y”
    Maximum MODIFICATION_DATE || TRANS_NO
```

Create entry in NDW_REFRESH_RUN_CONTROL.

```
Insert record into NDW_REFRESH_RUN_CONTROL
    Set REFRESH_ID_NO = next sequential number from NDW_MAXSEQNOS
    Set ACTIVE_STATUS = “Y”
    Set CFS_SNAPSHOTS_COMPLETE = “Y”
    Set MAX_TRIAL to maximum TRIAL_ID (for all account numbers)
    Set MAX_4700 to maximum TRIAL_ID for records with ACCOUNT_NO = 4700
    Set MAX_48XX to maximum TRIAL_ID for records with ACCOUNT_NO between 4800 and 4899
    Set MAX_49XX to maximum TRIAL_ID for records with ACCOUNT_NO between 4900 and 4999
    Set MAX_BUDGET to maximum BC.TRANS_NO found within the maximum MODIFICATION_DATE.
    Set MAX_BUDGET_MOD_DATE to maximum BC.MODIFICATION_DATE found
    Set MAX_LABOR to maximum GC.TRANS_NO found within the maximum MODIFICATION_DATE.
    Set MAX_LABOR_MOD_DATE to maximum GC.MODIFICATION_DATE found
    Set other fields to default values as shown in Appendix C – Support Table Record Layouts.
    Set NDW_LAST_MOD fields appropriately
If record cannot be inserted:
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW901_INITIATE.SQL”
        REFRESH_ID = next sequential number from NDW_MAXSEQNOS
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Record could not be inserted in NDW_REFRESH_RUN_CONTROL table.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

<u>FOR RESTARTS OR NEW REFRESH RUNS</u>

Update message in NDW_DEFAULTS table (this message may be queried by users to determine status of data warehouse or may be displayed on a menu/report.

```
Update NDW_DEFAULTS for record with
    ITEM_NAME = “NDW_REFRESH_MESSAGE”
        Set VALUE = “NDW REFRESH PROCESS IN PROGRESS – REFRESH ID 99999999”
    Set NDW_LAST_MOD fields appropriately
If record cannot be updated:
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW901_INITIATE.SQL”
        REFRESH_ID = next sequential number from NDW_MAXSEQNOS
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Could not update NDW_DEFAULTS table.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

Commit records to database.
```

Close Routine

Write all NDW_PROCESS_LOG records written for this routine to the .log file and close file.
Terminate processing.

4.1.2 Assign ACCS ID's (NDW902_REFRESH.SQL)

In order to improve the efficiency of performing the nightly refresh routines that update tables based on TRIAL data, the NOAA Data Mart will assign a unique ID for every ACCS combination (excluding Fund Code Fiscal Year and User Defined portions). This ID may also be used in lieu of comparing full ACCS strings between tables.

Routine Name

NDW902_REFRESH.SQL

Input:

This routine will be invoked through a regularly scheduled cron job and uses the following data as input:

- CAMS Data Warehouse Tables:
    - TRIAL (TR)
- NOAA Data Mart Tables:
    - NDW_ACCS_ID_CONTROL (NAIC)
    - NDW_ACCS_ID_MAP (NAIM)
    - NDW_MAXSEQNOS (NM)
    - NDW_DEFAULTS (ND)
    - NDW_REFRESH_RUN_CONTROL (NRRC)

Output:

This routine will produce the following return parameters, output files, database tables, and/or reports:

- NOAA Data Mart Tables:
    - NDW_ACCS_ID_CONTROL (NAIC)
    - NDW_ACCS_ID_MAP (NAIM)
    - NDW_REFRESH_RUN_CONTROL (NRRC)
    - NDW_MAXSEQNOS (NM)
    - NDW_PROCESS (NP)
- Reports:

- NDW902_RE999_PASS000.log (if database is available).
- NDW902_yyyymmdd.log (if database is not available).
- Only keep 10 versions of NDW902 log files in the directory.

Processing Logic:

```
Begin Routine
Write date/time started to .log file.
Write header information to .log file.

Determine refresh run.  There should be only one refresh run active at any time, although
multiple processes may be run in parallel.
Read REFRESH_ID_NO, NDW902_REFRESH_PASS_NO,
CFS_SNAPSHOTS_COMPLETE
    From NDW_REFRESH_RUN_CONTROL (NRRC)
    Where ACTIVE_STATUS = “Y”.

If more than one entry
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW902_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  More than one active
        NDW_REFRESH_RUN_CONTROL entry.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW902_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No active NDW_REFRESH_RUN_CONTROL
        entry found.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

Set v.REFRESH_PASS_NO = NDW902_REFRESH_PASS_NO + 1

If the snapshots of the CFS tables are not complete, terminate the job.
If NRRC.CFS_SNAPSHOTS_COMPLETE = “N”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW901_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Snapshots of CFS tables not complete.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

Determine maximum number of transactions to process and how often to commit.
```

```
Read VALUE(s)
    From NDW_DEFAULTS (ND) for the following ITEM_NAMEs
    ITEM_NAME = “NDW902_RECORDS_TO_PROCESS”
    ITEM_NAME = “NDW902_RECORDS_TO_COMMIT”
    ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”
If any entry cannot be found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW902_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Entry not found in NDW_DEFAULTS for
        ITEM_NAME: ??”.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG and terminate processing.

```
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW902_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Terminate processing.
End If END_NDW_BATCH_PROCESSES_FLAG
```

Determine beginning transaction number that should be used for selecting records from TRIAL for this routine.

```
Read LAST_TRANS_NO_PROCESSED from NDW_REFRESH_PARAMS (NRP)
    Where ROUTINE_NAME = “NDW902_REFRESH”
    And TABLE_NAME = “TRIAL”.
If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW902_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No entry found for routine in
        NDW_REFRESH_PARAMS table.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Write record to log table.

```
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW902_REFRESH.SQL”
    REFRESH_ID = NRRC.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    DATE_TIME = system date/time
    TRANS_NO = NRP.LAST_TRANS_NO_PROCESSED
    TABLE_NAME_PROCESSED = NRP.TABLE_NAME
    STEP_DESCR = “NDW-000TBD:  Routine started with TRIAL_ID > this number.”
    All other fields remain null.
```

Read TRIAL Records

Process only TRIAL records since the last refresh.

```
For each record within TRIAL (TR) where
    TRIAL_ID > NRP.LAST_TRANS_NO_PROCESSED and <
    NRP.LAST_TRANS_NO_PROCESSED + ND. NDW902_RECORDS_TO_PROCESS
```

Insert/Update NDW_ACCS_ID_CONTROL & NDW_ACCS_ID_MAP Records

Update NDW_ACCS_ID table based on TRIAL records

```
If TR.TRANS_SOURCE <> “BEGBAL”
    Match the TRIAL record (TR) ACCS values to an existing record in
    NDW_ACCS_ID_CONTROL (NAIC) – by ACCS.
```

Update existing record if found.

```
    If a match is found
        Insert a subordinate NDW_ACCS_ID_MAP (NAIM) record
        Set NAIM.TRIAL_ID to TR.TRIAL_ID
        Set NAIM.NDW_ACCS_ID to NAIC.NDW_ACCS_ID
        Set NAIM.NDW_LAST_MOD_DATE to system date
        Set NAIM.NDW_LAST_MOD_USER_NAME to user’s name
        Set NAIM.NDW_LAST_MOD_DEVICE_NAME to device
```

Add a new record if not found.

```
    Else (no match found)
        Create a new NDW_ACCS_ID_CONTROL (NAIC) record Appendix C – Support
        Table Record Layouts.
        Set NAIC.NDW_ACCS_ID to NAIC.NDW_ACCS_ID
        Set NAIC.NDW_LAST_MOD_DATE to system date
        Set NAIC.NDW_LAST_MOD_USER_NAME to user’s name
        Set NAIC.NDW_LAST_MOD_DEVICE_NAME to device

        Insert a subordinate NDW_ACCS_ID_MAP (NAIM) record
        Set NDW_ACCS_ID to next sequential number from NDW_MAXSEQNOS
        Where TABLE_NAME = “NDW_ACCS_ID”.
        Set NAIM.TRIAL_ID to TR.TRIAL_ID
        Set NAIM.NDW_ACCS_ID to NAIC.NDW_ACCS_ID
        Set NAIM.NDW_LAST_MOD_DATE to system date
        Set NAIM.NDW_LAST_MOD_USER_NAME to user’s name
        Set NAIM.NDW_LAST_MOD_DEVICE_NAME to device
```

Update control counts.

```
    Add 1 to v.RECORD_COUNT.
```

When all records have been processed, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG and commit records.

```
    If last TRIAL record
        Update NDW_REFRESH_PARAMS
            Where ROUTINE_NAME = “NDW902_REFRESH”
            Set LAST_TRANS_NO_PROCESSED = TR.TRIAL_ID
            Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
            Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
        Update NDW_REFRESH_RUN_CONTROL
            Set NDW902_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
            Set NDW902_MAX_VALUE_PROCESSED = TR.TRIAL_ID

        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW902_REFRESH.SQL”
            REFRESH_ID = NRRC.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            DATE_TIME = system date/time
            TRANS_NO = last TRIAL_ID read from TRIAL
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            RECORDS_PROCESSED = v.RECORD_COUNT
            AMOUNT_PROCESSED = .AMOUNT_PROCESSED
            STEP_DESCR = “NDW-000TBD:  Process completed.  Last TRIAL_ID
            processed updated”.
            All other fields remain null.
        Commit records to database
    End If last
```

After specified number of records have been processed, commit.

```
After each  ND.NDW902_RECORDS_TO_COMMIT records
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW902_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = TR.TRIAL_ID
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        STEP_DESCR = “NDW-000TBD:  Cumulative records committed to
        NDW_ACCS_ID_CONTROL table.”.
        All other fields remain null.
    Commit records to database.
End After Each
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG, commit records and terminate processing.

```
Read VALUE from NDW_DEFAULTS
    Where ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”.
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Update NDW_REFRESH_PARAMS
        Where ROUTINE_NAME = “NDW902_REFRESH”
        Set LAST_TRANS_NO_PROCESSED = TR.TRIAL_ID
        Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
        Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

    Update NDW_REFRESH_RUN_CONTROL
        Set NDW902_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
        Set NDW902_MAX_VALUE_PROCESSED = TR.TRIAL_ID

    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW902_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
```

```
            DATE_TIME = system date/time
            TRANS_NO = last TRIAL_ID read from TRIAL
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            RECORDS_PROCESSED = v.RECORD_COUNT
            AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
            STEP_DESCR = "NDW-000TBD:  Process terminated due to
            END_NDW_BATCH_PROCESSES_FLAG = "Y"".
            All other fields remain null.
        Commit records to database.
        Terminate processing.
    End If END_NDW_BATCH_PROCESSES_FLAG
End If TRANS_SOURCE <> "BEGBAL"
End Loop for Matching to NDW_ACCS_ID
```

<u>Close Routine</u>

```
Write all NDW_PROCESS_LOG records written for this routine to the .log file and close file.
If ND.END_NDW_BATCH_PROCESSED_FLAG = "N"
    And NRRC.NDW902_MAX_VALUE_PROCESSED < NRRC.MAX_TRIAL
    Restart the routine at the beginning for an additional pass until all records are processed.
Else
    Terminate processing.
```

4.2 NOAA Data Mart Summary Table Refresh (NDW001_REFRESH.SQL)

The NOAA Data Mart summary tables provide a summarized view of the detailed TRIAL transactions and Budget Operating Plan data from CFS. There are two summary tables that will be created based on TRIAL (NDW_GL_ACCT_SUMMARY and NDW_FIN_CAT_SUMMARY) and an additional table (NDW_BOP_SUMMARY) to summarize the budget data based on the BUDGET_CONTROL and BUDGET_DETAIL tables. The following subsections provide the detailed specifications for populating these tables. Appendix A – Summary Table Record Formats, provides the database table format for these tables.

4.2.1 Refresh NDW_GL_ACCT_SUMMARY Table (NDW001_REFRESH.SQL)

The purpose of this table is to provide summarized data by the distinct Standard General Ledger (SGL) account definition for reporting purposes. This table reduces the size of the table being queried for summary level reporting without losing accounting detail.

Routine Name

NDW001_REFRESH.SQL

Input:

This routine will be invoked through a regularly scheduled cron job and uses the following data as input:

- CAMS Data Warehouse Tables:
    - TRIAL (TR)
    - CHART (CH)
- NOAA Data Mart Tables:
    - NDW_GL_ACCT_SUMMARY (NGAS)
    - NDW_REFRESH_PARAMS (NRP)
    - NDW_DEFAULTS (ND)
    - NDW_REFRESH_RUN_CONTROL (NRRC)
    - NDW_ACCS_ID_MAP (NAIM)

Output:

This routine will produce the following return parameters, output files, database tables, and/or reports:

- NOAA Data Mart Tables:
    - NDW_GL_ACCT_SUMMARY (NGAS)
    - NDW_REFRESH_PARAMS (NRP)
    - NDW_REFRESH_RUN_CONTROL (NRRC)
    - NDW_PROCESS (NP)
- Reports:
    - NDW001_RE999_PASS999.log (if database is available).
    - NDW001_yyyymmdd.log (if database is not available).
    - Only keep 10 versions of NDW001 log files in the directory.

Processing Logic:

```
Begin Routine
Write date/time started to .log file.
Write header information to .log file.

Determine refresh run.  There should be only one refresh run active at any time, although
multiple processes may be run in parallel.
Read REFRESH_ID_NO, NDW001_REFRESH_PASS_NO,
CFS_SNAPSHOTS_COMPLETE
    From NDW_REFRESH_RUN_CONTROL (NRRC)
    Where ACTIVE_STATUS = "Y".

If more than one entry
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW001_REFRESH.SQL"
        DATE_TIME = system date/time
        STEP_DESCR = "NDW-000TBD:  More than one active
        NDW_REFRESH_RUN_CONTROL entry."
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW001_REFRESH.SQL"
        DATE_TIME = system date/time
        STEP_DESCR = "NDW-000TBD:  No active NDW_REFRESH_RUN_CONTROL
        entry found."
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

Set v.REFRESH_PASS_NO = NDW001_REFRESH_PASS_NO + 1
```

If the snapshots of the CFS tables are not complete, terminate the job.

```
If NRRC.CFS_SNAPSHOTS_COMPLETE = “N”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW001_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Snapshots of CFS tables not complete.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Determine maximum number of transactions to process and how often to commit.

```
Read VALUE(s)
    From NDW_DEFAULTS (ND) for the following ITEM_NAMEs
    ITEM_NAME = “NDW001_RECORDS_TO_PROCESS”
    ITEM_NAME = “NDW_GL_ACCT_SUMMARY_PRIVACY”
    ITEM_NAME = “NDW_GL_ACCT_SUMMARY_TABLE_ID”
    ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”
If any entry cannot be found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW001_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Entry not found in NDW_DEFAULTS for
        ITEM_NAME: ??”.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG and terminate processing.

```
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW001_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
End If END_NDW_BATCH_PROCESSES_FLAG
```

Determine beginning transaction number that should be used for selecting records from TRIAL for this routine.

```
Read LAST_TRANS_NO_PROCESSED from NDW_REFRESH_PARAMS (NRP)
    Where ROUTINE_NAME = “NDW001_REFRESH”
    And TABLE_NAME = “TRIAL”.
If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW001_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No entry found for routine in
        NDW_REFRESH_PARAMS table.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

▌ Write record to log table.
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW001_REFRESH.SQL”
    REFRESH_ID = NRRC.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    DATE_TIME = system date/time
    TRANS_NO = NRP.LAST_TRANS_NO_PROCESSED
    TABLE_NAME_PROCESSED = NRP.TABLE_NAME
    STEP_DESCR = “NDW-000TBD:  Routine started with TRIAL_ID > this number.”
    All other fields remain null.
```

Read TRIAL Records

```
▌ Process only TRIAL records since the last refresh.
For each record within TRIAL (TR) where
    TRIAL_ID > NRP.LAST_TRANS_NO_PROCESSED and <
    NRP.LAST_TRANS_NO_PROCESSED + ND. NDW001_RECORDS_TO_PROCESS
    And TR.TRANS_SOURCE <> “BEGBAL”
        Match TR.TRIAL_ID to NAIM.TRIAL_ID (to obtain NDW_ACCS_ID)

    Group by NDW_ACCS_ID, FUND_CODE_FISCAL_YEAR, GL_END_DATE,
    ACCOUNT_NO, and SUB_ACCOUNT_NO
```

Insert/Update NDW_GL_ACCT_SUMMARY Records

```
▌ Update NDW_GL_ACCT_SUMMARY table based on grouped TRIAL records
For each grouped record
    Match NAIM.NDW_ACCS_ID to NGAS.NDW_ACCS_ID,
        TR.FUND_CODE_FISCAL_YEAR to NGAS.FUND_CODE_FISCAL_YEAR,
        TR.GL_END_DATE to NGAS.GL_END_DATE,
        TR.ACCOUNT_NO to NGAS.ACCOUNT_NO
        TR.SUB_ACCOUNT_NO to NGAS.SUB_ACCOUNT_NO

  ▌ Update existing record if found.
    If a match is found update NDW_GL_ACCT_SUMMARY as follows:
        Add TR.DEBIT_AMOUNT to NGAS.DEBIT_AMOUNT
        Add TR.CREDIT_AMOUNT to NGAS.CREDIT_AMOUNT

      ▌ Compute Net Amount based on BALANCE_FLAG for the SGL account.
        If NGAS.BALANCE_FLAG = “DR”
            Compute v.NET_AMOUNT = TR.DEBIT_AMOUNT –
            TR.CREDIT_AMOUNT
        If NGAS.BALANCE_FLAG = “CR”
            Compute v.NET_AMOUNT = TR.CREDIT_AMOUNT –
            TR.DEBIT_AMOUNT
        If NGAS.BALANCE_FLAG = “DC”
            Do not update this field (should be 0).
        Add v.NET_AMOUNT to NGAS.NET_AMOUNT

      ▌ Add STAT_UNIT_QTY if the transaction is a labor record.
        If TR.ACCOUNT_NO between 4900 and 4999
            And TR.SUBSYSTEM_CODE = “GJ”
            And TR.TRANS_SOURCE = “GJ”
            And substr(TR.TRANS_DESCR,13,6) in (“NFC002”, “NFC005”, “NFC004”)
                Add TR.STAT_UNIT_QTY to NGAS.LABOR_HOURS

        Update LAST_MOD fields.
        Set NGAS.NDW_LAST_MOD_DATE to system date
        Set NGAS.NDW_LAST_MOD_USER_NAME to user’s name
        Set NGAS.NDW_LAST_MOD_DEVICE_NAME to device

    Add a new record if not found.
    Else (no match found)
        Set NDW_GL_ACCT_SUMMARY fields to TRIAL fields as reflected in Appendix
        A – Summary Table Record Formats.
        Set TRANS_TABLE_INDICATOR to
        ND.TABLE_ID_NDW_GL_ACCT_SUMMARY.
        Set NDW_TRANS_NO to next sequential number from NDW_MAXSEQNOS
        Where TABLE_NAME = “NDW_GL_ACCT_SUMMARY”.
        Set NDW_ACCS_ID to NAIM.NDW_ACCS_ID
        Set MONTH_CLOSED_FLAG = “N”.
        Set PRIVACY_ACT_APPLIES to ND.PRIVACY_NDW_GL_ACCT_SUMMARY.


        Derive FISCAL_MONTH field from GL_END_DATE
        If TR.GL_END_DATE month > 9
            Set NGAS.FISCAL_MONTH = TR.GL_END_DATE month - 9
        Else
            Set NGAS.FISCAL_MONTH = TR.GL_END_DATE month + 3

        Set BALANCE_FLAG based on ACCOUNT_NO/SUB_ACCOUNT_NO in
        CHART
        Set NGAS.BALANCE_FLAG to CH.BALANCE_FLAG from CHART (CH)
            Where CH.ACCOUNT_NO = TR.ACCOUNT_NO
            And CH.SUB_ACCOUNT_NO = TR.SUB_ACCOUNT_NO

        Compute Net Amount based on BALANCE_FLAG for the SGL account.
        If NGAS.BALANCE_FLAG = “DR”
            Set NGAS.NET_AMOUNT = TR.DEBIT_AMOUNT –
            TR.CREDIT_AMOUNT
        If NGAS.BALANCE_FLAG = “CR”
            Set NGAS.NET_AMOUNT = TR.CREDIT_AMOUNT –
            TR.DEBIT_AMOUNT
        If NGAS.BALANCE_FLAG = “DC”
            Set NGAS.NET_AMOUNT to 0.

        Add STAT_UNIT_QTY if the transaction is a labor record.
        If TR.ACCOUNT_NO between 4900 and 4999
            And TR.SUBSYSTEM_CODE = “GJ”
            And TR.TRANS_SOURCE = “GJ”
            And substr(TR.TRANS_DESCR,13,6) in (“NFC002”,”NFC005”,”NFC004”)
                Set TR.STAT_UNIT_QTY to NGAS.LABOR_HOURS

        Populate both CREATION and LAST_MOD fields.
        Set NDW_CREATION_DATE to system date
        Set NDW_CREATION_USER_NAME to user’s name
        Set NDW_CREATION_DEVICE_NAME to device

        Set NDW_LAST_MOD_DATE to system date
        Set NDW_LAST_MOD_USER_NAME to user’s name
        Set NDW_LAST_MOD_DEVICE_NAME to device

        Set all Line Office extract fields to null
```

Update control counts.

```
Add 1 to v.RECORD_COUNT.
Add (TR.DEBIT_AMOUNT – TR.CREDIT_AMOUNT) to v.AMOUNT_PROCESSED.
```

When all records have been processed, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG and commit records.

```
If last TRIAL record
    Update NDW_REFRESH_PARAMS
        Where ROUTINE_NAME = “NDW001_REFRESH”
        Set LAST_TRANS_NO_PROCESSED = TR.TRIAL_ID
        Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
        Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

    Update NDW_REFRESH_RUN_CONTROL
        Set NDW001_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
        Set NDW001_MAX_VALUE_PROCESSED = TR.TRIAL_ID

    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW001_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = last TRIAL_ID read from TRIAL
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        AMOUNT_PROCESSED = .AMOUNT_PROCESSED
        STEP_DESCR = “NDW-000TBD:  Process completed.  Last TRIAL_ID
        processed updated”.
        All other fields remain null.
    Commit records to database
End If last
```

After specified number of records have been processed, commit.

```
After each  ND.NDW001_RECORDS_TO_COMMIT records
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW001_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = TR.TRIAL_ID
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
        STEP_DESCR = “NDW-000TBD:  Cumulative records committed to
        NDW_GL_ACCT_SUMMARY table.”.
        All other fields remain null.
    Commit records to database.
    End After Each
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG, commit records and terminate processing.

```
    Read VALUE from NDW_DEFAULTS
        Where ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”.
    If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
        Update NDW_REFRESH_PARAMS
            Where ROUTINE_NAME = “NDW001_REFRESH”
            Set LAST_TRANS_NO_PROCESSED = TR.TRIAL_ID
            Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
            Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

        Update NDW_REFRESH_RUN_CONTROL
            Set NDW001_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
            Set NDW001_MAX_VALUE_PROCESSED = TR.TRIAL_ID

        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW001_REFRESH.SQL”
            REFRESH_ID = NRRC.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            DATE_TIME = system date/time
            TRANS_NO = last TRIAL_ID read from TRIAL
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            RECORDS_PROCESSED = v.RECORD_COUNT
            AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
            STEP_DESCR = “NDW-000TBD:  Process terminated due to
            END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
            All other fields remain null.
        Commit records to database.
        Terminate processing.
    End If END_NDW_BATCH_PROCESSES_FLAG
End If TRANS_SOURCE <> “BEGBAL”
End Loop for Matching to NDW_GL_ACCT_SUMMARY
```

<u>Close Routine</u>

Write all NDW_PROCESS_LOG records written for this routine to the .log file and close file.

```
If ND.END_NDW_BATCH_PROCESSED_FLAG = “N”
    And NRRC.NDW001_MAX_VALUE_PROCESSED < NRRC.MAX_TRIAL
    Restart the routine at the beginning for an additional pass until all records are processed.
Else
    Terminate processing.
```

4.2.2 Refresh NDW_FIN_CAT_SUMMARY Table (NDW002_REFRESH.SQL)

The purpose of this table is to provide summarized data by NOAA-oriented financial categories for Line/Field Office extract and reporting purposes. This table further reduces the volume of data being queried as well as providing user-oriented definitions.

Routine Name

NDW002_REFRESH.SQL

Input:

This routine will be invoked through a regularly scheduled cron job and uses the following data as input:

- CAMS Data Warehouse Tables:
  - TRIAL (TR)
- NOAA Data Mart Tables:
  - NDW_FIN_CAT_SUMMARY (NFCS)
  - NDW_REFRESH_PARAMS (NRP)
  - NDW_DEFAULTS (ND)
  - NDW_REFRESH_RUN_CONTROL (NRRC)
  - NDW_FIN_CAT_DEF_DETAIL (NFCDD)
  - NDW_ACCS_ID_MAP (NAIM)

Output:

This routine will produce the following return parameters, output files, database tables, and/or reports:

- NOAA Data Mart Tables:
  - NDW_FIN_CAT_SUMMARY (NFCS)
  - NDW_REFRESH_PARAMS (NRP)
  - NDW_REFRESH_RUN_CONTROL (NRRC)
  - NDW_PROCESS (NP)
- Reports:

- NDW002_RE999_PASS999.log (if database is available).
- NDW002_yyyymmdd.log (if database is not available).
- Only keep 10 versions of NDW002 log files in the directory.

Processing Logic:

Begin Routine

Write date/time started to .log file.

```
Write header information to .log file.
```

Determine refresh run. There should be only one refresh run active at any time, although multiple processes may be run in parallel.

```
Read REFRESH_ID_NO, NDW002_REFRESH_PASS_NO,
CFS_SNAPSHOTS_COMPLETE
    From NDW_REFRESH_RUN_CONTROL (NRRC)
    Where ACTIVE_STATUS = “Y”.

If more than one entry
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW002_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  More than one active
        NDW_REFRESH_RUN_CONTROL entry.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW002_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No active NDW_REFRESH_RUN_CONTROL
        entry found.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

Set v.REFRESH_PASS_NO = NDW002_REFRESH_PASS_NO + 1
```

If the snapshots of the CFS tables are not complete, terminate the job.

```
If NRRC.CFS_SNAPSHOTS_COMPLETE = “N”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW002_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Snapshots of CFS tables not complete.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Determine maximum number of transactions to process and how often to commit.

```
Read VALUE(s)
```

```
    From NDW_DEFAULTS (ND) for the following ITEM_NAMEs
    ITEM_NAME = “NDW002_RECORDS_TO_PROCESS”
    ITEM_NAME = “NDW_FIN_CAT_SUMMARY_PRIVACY”
    ITEM_NAME = “NDW_FIN_CAT_SUMMARY_TABLE_ID”
    ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”
If any entry cannot be found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW002_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Entry not found in NDW_DEFAULTS for
        ITEM_NAME: ??”.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG and terminate processing.

```
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW002_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Update NDW_REFRESH_RUN_CONTROL where
        Set REFRESH_STOPPED_BY_OPERATOR = “Y”
    Terminate processing.
End If END_NDW_BATCH_PROCESSES_FLAG
```

Determine beginning transaction number that should be used for selecting records from TRIAL for this routine.

```
Read LAST_TRANS_NO_PROCESSED from NDW_REFRESH_PARAMS (NRP)
    Where ROUTINE_NAME = “NDW002_REFRESH”
    And TABLE_NAME = “TRIAL”.
If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW002_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No entry found for routine in
        NDW_REFRESH_PARAMS table.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Write record to log table.

```
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW002_REFRESH.SQL”
    REFRESH_ID = NRRC.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    DATE_TIME = system date/time
    TRANS_NO = NRP.LAST_TRANS_NO_PROCESSED
    TABLE_NAME_PROCESSED = NRP.TABLE_NAME
    STEP_DESCR = “NDW-000TBD:  Routine started with TRIAL_ID > this number.”
    All other fields remain null.
```

Read TRIAL Records

- ***Process only TRIAL records since the last refresh.***

```
For each record within TRIAL (TR) where
    TRIAL_ID > NRP.LAST_TRANS_NO_PROCESSED and <
    NRP.LAST_TRANS_NO_PROCESSED + ND. NDW002_RECORDS_TO_PROCESS
    And TR.TRANS_SOURCE <> “BEGBAL”
        Match TR.TRIAL_ID to NAIM.TRIAL_ID (to obtain NDW_ACCS_ID)

    Group by NDW_ACCS_ID, FUND_CODE_FISCAL_YEAR, GL_END_DATE,
    ACCOUNT_NO, and SUB_ACCOUNT_NO
```

Insert/Update NDW_FIN_CAT_SUMMARY Records

For each grouped TRIAL (TR) record:

- ***Determine Financial Category(s) in which this TRIAL record applies.***

```
Read FIN_CAT_NO, FIN_CAT_ID, FIN_CAT_GROUP_FLAG, BALANCE_FLAG
    from  NDW_FIN_CAT_DET_DETAIL (NFCDD) records
    Where TR.ACCOUNT_NO = NFCDD.ACCOUNT_NO
    And TR.SUB_ACCOUNT_NO = NFCDD.SUB_ACCOUNT_NO
    And NFCDD.ACTIVE_STATUS = “Y”
    And GL_END_DATE is between NFCDD.START_DATE and NFCDD.END_DATE
```

If grouped TRIAL record does not fall within any of the Financial Category definitions, do not process. Proceed to next grouped record.

- ***Update NDW_FIN_CAT_SUMMARY table***

```
    Match NAIM.NDW_ACCS_ID to NFCS.NDW_ACCS_ID
    And TR.FUND_CODE_FISCAL_YEAR to NFCS.FUND_CODE_FISCAL_YEAR
    And TR.GL_END_DATE to NFCS.GL_END_DATE
```

  - ***Update existing record if found.***

```
    If a match is found:
        For each NFCDD.FIN_CAT_NO found
            Where TR.ACCOUNT_NO = NFCDD.ACCOUNT_NO
            And TR.SUB_ACCOUNT_NO = NFCDD.SUB_ACCOUNT_NO
```

    - ***Compute Net Amount based on BALANCE_FLAG for the SGL account.***

```
            If NFCDD.BALANCE_FLAG = “DR”
                Compute v.NET_AMOUNT = TR.DEBIT_AMOUNT –
                TR.CREDIT_AMOUNT
            Else if NFCDD.BALANCE_FLAG = “CR”
                Compute v.NET_AMOUNT = TR.CREDIT_AMOUNT –
                TR.DEBIT_AMOUNT
            Else set v.NET_AMOUNT = 0

            Add v.NET_AMOUNT to NFCS.FIN_CAT_”FIN_CAT_NO”_AMOUNT
```

    - ***Add STAT_UNIT_QTY if the transaction is a labor record.***

```
            If TR.ACCOUNT_NO between 4900 and 4999
                And TR.SUBSYSTEM_CODE = “GJ”
```

```
        And TR.TRANS_SOURCE = “GJ”
        And substr (TR.TRANS_DESCR,13,6) in (“NFC002”, “NFC005”,
        “NFC004”)
            Add TR.STAT_UNIT_QTY to
            NFCS.FIN_CAT_”FIN_CAT_NO”_LABOR_HOURS
    End For Each FIN_CAT_NO

  ▌ Update LAST_MOD fields.
    Set NFCS.NDW_LAST_MOD_DATE to system date
    Set NFCS.NDW_LAST_MOD_USER_NAME to user’s name
    Set NFCS.NDW_LAST_MOD_DEVICE_NAME to device

▌ Add a new record if not found.
Else (no match found)
    Set NDW_FIN_CAT_SUMMARY fields to TRIAL fields as reflected in Appendix
    A – Summary Table Record Formats.

    Set TRANS_TABLE_INDICATOR to
    ND.TABLE_ID_NDW_FIN_CAT_SUMMARY.

    Set NDW_TRANS_NO to next sequential number from NDW_MAXSEQNOS
    where TABLE_NAME = “NDW_FIN_CAT_SUMMARY”.

    Set MONTH_CLOSED_FLAG = “N”.

    Set PRIVACY_ACT_APPLIES to ND.PRIVACY_NDW_FIN_CAT_SUMMARY.

  ▌ Derive FISCAL_MONTH field from GL_END_DATE
    If TR.GL_END_DATE month > 9
        Set NFCS.FISCAL_MONTH = TR.GL_END_DATE month - 9
    Else
        Set NFCS.FISCAL_MONTH = TR.GL_END_DATE month + 3

    For each NFCDD.FIN_CAT_NO found
        Where TR.ACCOUNT_NO = NFCDD.ACCOUNT_NO
            And TR.SUB_ACCOUNT_NO = NFCDD.SUB_ACCOUNT_NO

          ▌ Compute Net Amount based on BALANCE_FLAG for the SGL account.
            If NFCDD.BALANCE_FLAG = “DR”
                Compute v.NET_AMOUNT = TR.DEBIT_AMOUNT –
                TR.CREDIT_AMOUNT
            Else if NFCDD.BALANCE_FLAG =  “CR”
                Compute v.NET_AMOUNT = TR.CREDIT_AMOUNT –
                TR.DEBIT_AMOUNT
            Else Set v.NET_AMOUNT = 0

            Set NFCS.FIN_CAT_”FIN_CAT_NO”_AMOUNT to v.NET_AMOUNT

          ▌ Add STAT_UNIT_QTY if the transaction is a labor record.
            If TR.ACCOUNT_NO between 4900 and 4999
                And TR.SUBSYSTEM_CODE = “GJ”
                And TR.TRANS_SOURCE = “GJ”
                And substr (TR.TRANS_DESCR,13,6) in (“NFC002”, “NFC005”,
                “NFC004”)
```

```
                    Set NFCS.FIN_CAT_"FIN_CAT_NO"_LABOR_HOURS to
                    TR.STAT_UNIT_QTY
    End For Each FIN_CAT_NO

    Populate both CREATION and LAST_MOD fields.
    Set NDW_CREATION_DATE to system date
    Set NDW_CREATION_USER_NAME to user's name
    Set NDW_CREATION_DEVICE_NAME to device

    Set NDW_LAST_MOD_DATE to system date
    Set NDW_LAST_MOD_USER_NAME to user's name
    Set NDW_LAST_MOD_DEVICE_NAME to device

    Set all Line Office extract fields to null

Update control counts.
Add 1 to v.RECORD_COUNT.
Add (TR.DEBIT_AMOUNT – TR.CREDIT_AMOUNT) to v.AMOUNT_PROCESSED.

When all records have been processed, update the NDW_REFRESH_PARAMS,
NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG and commit records.
If last TRIAL record
    Update NDW_REFRESH_PARAMS
        Where ROUTINE_NAME = "NDW002_REFRESH"
        Set LAST_TRANS_NO_PROCESSED = TR.TRIAL_ID
        Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
        Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

    Update NDW_REFRESH_RUN_CONTROL
        Set NDW002_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
        Set NDW002_MAX_VALUE_PROCESSED = TR.TRIAL_ID

    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW002_REFRESH.SQL"
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = last TRIAL_ID read from TRIAL
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        AMOUNT_PROCESSED = .AMOUNT_PROCESSED
        STEP_DESCR = "NDW-000TBD:  Process completed.  Last TRIAL_ID
        processed updated".
        All other fields remain null.
    Commit records to database
End If last

After specified number of records have been processed, commit.
After each  ND.NDW002_RECORDS_TO_COMMIT records
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW002_REFRESH.SQL"
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = TR.TRIAL_ID
```

```
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            RECORDS_PROCESSED = v.RECORD_COUNT
            AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
            STEP_DESCR = “NDW-000TBD:  Cumulative records committed to
            NDW_FIN_CAT_SUMMARY table.”.
            All other fields remain null.
        Commit records to database.
    End After Each
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG, commit records and terminate processing.

```
    Read VALUE from NDW_DEFAULTS
        Where ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”.
    If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
        Update NDW_REFRESH_PARAMS
            Where ROUTINE_NAME = “NDW002_REFRESH”
            Set LAST_TRANS_NO_PROCESSED = TR.TRIAL_ID
            Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
            Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

        Update NDW_REFRESH_RUN_CONTROL
            Set NDW002_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
            Set NDW002_MAX_VALUE_PROCESSED = TR.TRIAL_ID

        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW002_REFRESH.SQL”
            REFRESH_ID = NRRC.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            DATE_TIME = system date/time
            TRANS_NO = last TRIAL_ID read from TRIAL
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            RECORDS_PROCESSED = v.RECORD_COUNT
            AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
            STEP_DESCR = “NDW-000TBD:  Process terminated due to
            END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
            All other fields remain null.
        Commit records to database.
        Terminate processing.
    End If END_NDW_BATCH_PROCESSES_FLAG
End for each grouped TRIAL record
End Loop for Matching to NDW_FIN_CAT_SUMMARY
```

Close Routine

```
Write all NDW_PROCESS_LOG records written for this routine to the .log file and close
file.
If ND.END_NDW_BATCH_PROCESSED_FLAG = “N”
    And NRRC.NDW002_MAX_VALUE_PROCESSED < NRRC.MAX_TRIAL
    Restart the routine at the beginning for an additional pass until all records are processed.
Else
    Terminate processing.
```

4.2.3 Refresh NDW_BOP_SUMMARY Table (NDW003_REFRESH.SQL)

The purpose of this table is to provide summarized Budget Operating Plan data for Line/Field Office extract and reporting purposes.

Routine Name

NDW003_REFRESH.SQL

Input:

This routine will be invoked through a regularly scheduled cron job and uses the following data as input:

- CAMS Data Warehouse Tables:
    - BUDGET_CONTROL (BC)
    - BUDGET_DETAIL (BD)
- NOAA Data Mart Tables:
    - NDW_BOP_SUMMARY (NBS)
    - NDW_REFRESH_PARAMS (NRP)
    - NDW_DEFAULTS (ND)
    - NDW_REFRESH_RUN_CONTROL (NRRC)

Output:

This routine will produce the following return parameters, output files, database tables, and/or reports:

- NOAA Data Mart Tables:
    - NDW_BOP_SUMMARY (NBS)
    - NDW_REFRESH_PARAMS (NRP)
    - NDW_REFRESH_RUN_CONTROL (NRRC)
    - NDW_PROCESS (NP)
- Reports:
    - NDW003_RE999_PASS999.log (if database is available).
    - NDW003_yyyymmdd.log (if database is not available).
    - Only keep 10 versions of NDW003 log files in the directory.

Processing Logic:

Begin Routine

Write date/time started to .log file.

```
Write header information to .log file.
```

Determine refresh run. There should be only one refresh run active at any time, although multiple processes may be run in parallel.

```
Read REFRESH_ID_NO, NDW003_REFRESH_PASS_NO,
CFS_SNAPSHOTS_COMPLETE
    From NDW_REFRESH_RUN_CONTROL (NRRC)
    Where ACTIVE_STATUS = “Y”.

If more than one entry
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW003_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  More than one active
        NDW_REFRESH_RUN_CONTROL entry.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW003_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No active NDW_REFRESH_RUN_CONTROL
        entry found.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

Set v.REFRESH_PASS_NO = NDW003_REFRESH_PASS_NO + 1
```

If the snapshots of the CFS tables are not complete, terminate the job.

```
If NRRC.CFS_SNAPSHOTS_COMPLETE = “N”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW003_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Snapshots of CFS tables not complete.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Determine maximum number of transactions to process and how often to commit.

```
Read VALUE(s)
    From NDW_DEFAULTS (ND) for the following ITEM_NAMEs
    ITEM_NAME = “NDW003_RECORDS_TO_COMMIT”
    ITEM_NAME = “NDW_BOP_SUMMARY_PRIVACY”
    ITEM_NAME = “NDW_BOP_SUMMARY_TABLE_ID”
    ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”
If any entry cannot be found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW003_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Entry not found in NDW_DEFAULTS for
        ITEM_NAME: ??”.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG and terminate processing.

```
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW003_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
End END_NDW_BATCH_PROCESSES_FLAG
```

Determine last modification date and trans_no fields that should be used for selecting records from BUDGET_CONTROL and BUDGET_DETAIL for this routine.

```
Read LAST_TRANS_NO_PROCESSED, BEGIN_MOD_DATE
    From NDW_REFRESH_PARAMS (NRP)
    Where ROUTINE_NAME = “NDW003_REFRESH”
    And TABLE_NAME = “BUDGET_CONTROL”.
If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW003_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No entry found for routine in
        NDW_REFRESH_PARAMS table.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Write record to log table.

```
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW003_REFRESH.SQL”
    REFRESH_ID = NRRC.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    DATE_TIME = system date/time
    TRANS_NO = NRP.LAST_TRANS_NO_PROCESSSED
    MODIFICATION_DATE = NRP.BEGIN_MOD_DATE
    TABLE_NAME_PROCESSED = NRP.TABLE_NAME
    STEP_DESCR = “NDW-000TBD:  Routine started with key fields > this date and
    number.”
    All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Read BUDGET_CONTROL & BUDGET_DETAIL Records

Process only BUDGET_CONTROL/DETAIL records since the last refresh.

Note that because budget records can be approved out of sequence and there is no unique identifier for the BUDGET_DETAIL record, there is no RECORDS_TO_PROCESS field associated with these records. All newly approved BUDGET_CONTROL and BUDGET_DETAIL records are selected for processing.

```
Read BUDGET_CONTROL & BUDGET_DETAIL records
    Where BC.APPROVED_FLAG = "Y"
    And BC.MODIFICATION_DATE ||[1] BC.TRANS_NO > NRP.BEGIN_MOD_DATE ||
    NRP.LAST_TRANS_NO_PROCESSED

    Order by BC.MODIFICATION_DATE, BC.TRANS_NO
```

Insert/Update NDW_BOP_SUMMARY Records

Update NDW_BOP_SUMMARY table

```
    Match:
        BC.FISCAL_YEAR to NBS.FISCAL_YEAR
        BD.BUDGET_MONTH to NBS.FISCAL_MONTH
        BC.FUND_CODE to NBS.FUND_CODE
        BC.BUREAU_CODE to NBS.BUREAU_CODE
        BC.ORG1_CODE to NBS.ORG1_CODE
        BC.ORG2_CODE to NBS.ORG2_CODE
        BC.ORG3_CODE to NBS.ORG3_CODE
        BC.ORG4_CODE to NBS.ORG4_CODE
        BC.ORG5_CODE to NBS.ORG5_CODE
        BC.ORG6_CODE to NBS.ORG6_CODE
        BC.ORG7_CODE to NBS.ORG7_CODE
        BC.PROGRAM1_CODE to NBS.PROGRAM1_CODE
        BC.PROGRAM2_CODE to NBS.PROGRAM2_CODE
        BC.PROGRAM3_CODE to NBS.PROGRAM3_CODE
        BC.PROGRAM4_CODE to NBS.PROGRAM4_CODE
        BC.PROJECT_CODE to NBS.PROJECT_CODE
        BC.TASK_CODE to NBS.TASK_CODE
        BD.OBJECT1_CODE to NBS_OBJECT1_CODE
        BD.OBJECT2_CODE to NBS_OBJECT2_CODE
        BD.OBJECT3_CODE to NBS_OBJECT3_CODE
        BD.OBJECT4_CODE to NBS_OBJECT4_CODE
```

Update existing record if found.

```
    If a match is found update NDW_BOP_SUMMARY as follows:
        Add BD.AMOUNT to NBS.BUDGET_AMOUNT
        Add BD.STAT_UNIT_QTY to NBS.LABOR_FTE
```

Update LAST_MOD fields.

```
        Set NBS.NDW_LAST_MOD_DATE to system date
```

[1] The symbol || indicates that the fields should be concatenated (strung together).

```
    Set NBS.NDW_LAST_MOD_USER_NAME to user's name
    Set NBS.NDW_LAST_MOD_DEVICE_NAME to device

Add a new record if not found.
Else (no match found)
    Set NDW_BOP_SUMMARY fields to BUDGET_CONTROL/BUDGET_DETAIL
    fields as reflected in Appendix A – Summary Table Record Formats.

    Set TRANS_TABLE_INDICATOR to ND.TABLE_ID_NDW_BOP_SUMMARY.
    Set NDW_TRANS_NO to next sequential number from NDW_MAXSEQNOS
    Where TABLE_NAME = "NDW_BOP_SUMMARY"
    Set NBS.BUDGET_AMOUNT to BD.AMOUNT
    Set NBS.LABOR_FTE to BD. STAT_UNIT_QTY
    Set MONTH_CLOSED_FLAG = "N".
    Set PRIVACY_ACT_APPLIES to ND.PRIVACY_NDW_BOP_SUMMARY.

    Populate both CREATION and LAST_MOD fields.
    Set NDW_CREATION_DATE to system date
    Set NDW_CREATION_USER_NAME to user's name
    Set NDW_CREATION_DEVICE_NAME to device

    Set NDW_LAST_MOD_DATE to system date
    Set NDW_LAST_MOD_USER_NAME to user's name
    Set NDW_LAST_MOD_DEVICE_NAME to device

    Set all Line Office extract fields to null
```

For Updates or Inserts:

```
Update control counts.
Add 1 to v.RECORD_COUNT.
Add BD.AMOUNT to v.AMOUNT_PROCESSED.

When all records have been processed, update the NDW_REFRESH_PARAMS,
NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG and commit records.
If last BUDGET_DETAIL record
    Update NDW_REFRESH_PARAMS
        Where ROUTINE_NAME = "NDW003_REFRESH"
        Set LAST_TRANS_NO_PROCESSED = BC.TRANS_NO
        Set BEGIN_MOD_DATE = BC.MODIFICATION_DATE
        Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
        Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

    Update NDW_REFRESH_RUN_CONTROL
        Set NDW003_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
        Set NDW003_MAX_VALUE_PROCESSED = BC.TRANS_NO
        Set NDW003_LAST_MOD_DATE = BC.MODIFICATION_DATE

    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW003_REFRESH.SQL"
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = BC.TRANS_NO
        MODIFICATION_DATE = BC.MODIFICATION_DATE
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        AMOUNT_PROCESSED = .AMOUNT_PROCESSED
        STEP_DESCR = “NDW-000TBD:  Process completed.  Last approved
        BUDGET_CONTROL record with this key processed”.
        All other fields remain null.
    Commit records to database
End If last
```

After specified number of records has been processed, commit.

```
After each ND.NDW003_RECORDS_TO_COMMIT records

    Check to see if the next BUDGET_DETAIL record has the same TRANS_NO as the
    last one processed.
```

Because the BUDGET_DETAIL record does not have a unique key, the BUDGET_DETAIL records must be processed until the BUDGET_CONTROL TRANS_NO key changes.

```
    If so, continue processing the records until the next record’s TRANS_NO changes or
    no next record.

    Else

    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW003_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = BC.TRANS_NO
        MODIFICATION_DATE = BC.MODIFICATION_DATE
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
        STEP_DESCR = “NDW-000TBD:  Cumulative records committed to
        NDW_BOP_SUMMARY table.”.
        All other fields remain null.
    Commit records to database.
End After Each
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG, commit records and terminate processing.

```
Read VALUE from NDW_DEFAULTS
    Where ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”.
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Update NDW_REFRESH_PARAMS
        Where ROUTINE_NAME = “NDW003_REFRESH”
        Set LAST_TRANS_NO_PROCESSED = BC.TRANS_NO
        Set BEGIN_MOD_DATE = BC.MODIFICATION_DATE
        Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
        Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

    Update NDW_REFRESH_RUN_CONTROL
        Set NDW003_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
```

```
            Set NDW003_MAX_VALUE_PROCESSED = BC.TRANS_NO
            Set NDW003_LAST_MOD_DATE = BC.MODIFICATION_DATE

        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW003_REFRESH.SQL”
            REFRESH_ID = NRRC.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            DATE_TIME = system date/time
            TRANS_NO = BC.TRANS_NO
            MODIFICATION_DATE = BC.MODIFICATION_DATE
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            RECORDS_PROCESSED = v.RECORD_COUNT
            AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
            STEP_DESCR = “NDW-000TBD:  Process terminated due to
            END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
            All other fields remain null.
        Commit records to database.
        Terminate processing.

    End If END_NDW_BATCH_PROCESSES_FLAG
End Loop for Matching to NDW_BOP_SUMMARY
```

Close Routine

Write all NDW_PROCESS_LOG records written for this routine to the .log file and close file.

Terminate processing.

4.2.4 Update Summary Tables to Reflect Accounting Period Status

Summary level tables are updated on a nightly basis. The amounts within a record may be updated multiple times during the monthly accounting period, until that period is closed. Because the data is not static until the accounting period is closed, it is useful to know the status of the month in order to better interpret the data. This routine updates data warehouse summary table records to indicate that the general ledger accounting period (month) is closed.

Note that this job should not be run until after NDW001_REFRESH.SQL and NDW002_REFRESH.SQL have completed.

Routine Name

NDW050.SQL

Input:

This routine will be invoked through a regularly scheduled cron job and uses the following data as input:

- CAMS Data Warehouse Tables:
  - APERIOD (A)
- NOAA Data Mart Tables:
  - NDW_ACCT_PERIOD_STATUS (NAPS)
  - NDW_GL_ACCT_SUMMARY (NGAS)
  - NDW_FIN_CAT_SUMMARY (NFCS)

Output:

This routine will produce the following return parameters, output files, database tables, and/or reports:

- NOAA Data Mart Tables:
  - NDW_ACCT_PERIOD_STATUS (NAPS)
  - NDW_GL_ACCT_SUMMARY (NGAS)
  - NDW_FIN_CAT_SUMMARY (NFCS)
  - NDW_PROCESS (NP)
- Reports:
  - NDW050_yyyymmdd.log

Processing Logic:

Begin Routine

Write date/time started to .log file.

```
Write header information to .log file.
```

Determine refresh run. There should be only one refresh run active at any time, although multiple processes may be run in parallel.

Unlike the refresh routines, the NDW050 routine can be run outside of the normal job stream if necessary. Therefore, if an active NDW_REFRESH_RUN_CONTROL is not found, an informational message should be written and the REFRESH_ID_NO and PASS_NO in the NDW_PROCESS_LOG will be set to 0.

```
Read REFRESH_ID_NO
    From NDW_REFRESH_RUN_CONTROL (NRRC)
    Where ACTIVE_STATUS = "Y".

If more than one entry
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW050.SQL"
        DATE_TIME = system date/time
        STEP_DESCR = "NDW-000TBD:  More than one active
        NDW_REFRESH_RUN_CONTROL entry."
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW050.SQL"
        DATE_TIME = system date/time
        REFRESH_ID = 0
        PASS_NO = 0
        STEP_DESCR = "NDW-000TBD:  No active NDW_REFRESH_RUN_CONTROL
        entry found."
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Do not terminate processing.
    Set v.REFRESH_ID_NO = 0
    Set v.REFRESH_PASS_NO = 0

Else if a single entry is found
    Set v.REFRESH_ID_NO = NRRC.REFRESH_ID_NO
    Set v.REFRESH_PASS_NO = NRRC.REFRESH_PASS_NO
```

Determine how often to commit.

```
Read VALUE(s)
    From NDW_DEFAULTS (ND) for the following ITEM_NAMEs
    ITEM_NAME = "NDW050_RECORDS_TO_COMMIT"
    ITEM_NAME = "END_NDW_BATCH_PROCESSES_FLAG"

If any entry cannot be found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW050.SQL"
        DATE_TIME = system date/time
        REFRESH_ID = v.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        STEP_DESCR = “NDW-000TBD:  Entry not found in NDW_DEFAULTS for
        ITEM_NAME: ??”.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG and terminate processing.

```
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW050.SQL”
        DATE_TIME = system date/time
        REFRESH_ID = v.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Update NDW_ACCT_PERIOD_STATUS Table

Determine last modification date that should be used for selecting records from APERIOD for this routine.

```
Read BEGIN_MOD_DATE from NDW_REFRESH_PARAMS (NRP)
    Where ROUTINE_NAME = “NDW050”
    And TABLE_NAME = “APERIOD”.
If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW050.SQL”
        DATE_TIME = system date/time
        REFRESH_ID = v.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        STEP_DESCR = “NDW-000TBD:  No entry found for routine in
        NDW_REFRESH_PARAMS table.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Write record to log table.

```
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW050.SQL”
    DATE_TIME = system date/time
    REFRESH_ID = v.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    TABLE_NAME_PROCESSED = “APERIOD”
    STEP_DESCR = “NDW-000TBD:  Process to update NDW_ACCT_PERIOD_STATUS
    table started.”
    All other fields remain null.

Read all APERIOD (A) records
    Where A.MODIFICATION_DATE > NRP.BEGIN_MOD_DATE
```

```
For each APERIOD record read
    Match A.GL_BEGIN_DATE to NAPS. GL_BEGIN_DATE
    Match A.GL_END_DATE to NAPS.GL_END_DATE

    If a match is found
        Update NDW_ACCT_PERIOD_STATUS record:
            Set NAPS.GL_STATUS to A.GL_STATUS
            Set NDW_LAST_MOD_DATE to system date
            Set NDW_LAST_MOD_USER_NAME to user’s name
            Set NDW_LAST_MOD_DEVICE_NAME to device
        Add 1 to v.RECORD_COUNT

    If a match is not found
        Insert new NDW_ACCT_PERIOD_STATUS:
            Set NAPS.GL_BEGIN_DATE to A.GL_BEGIN_DATE
            Set NAPS.GL_END_DATE to A.GL_END_DATE
            Set NAPS.GL_STATUS to A.GL_STATUS
            Set NDW_LAST_MOD_DATE to system date
            Set NDW_LAST_MOD_USER_NAME to user’s name
            Set NDW_LAST_MOD_DEVICE_NAME to device
        Add 1 to v.RECORD_COUNT

End For Each APERIOD Record
```

Write record to log table.

```
When all APERIOD records are processed

    Update NDW_REFRESH_PARAMS (NRP) entry
        Where ROUTINE_NAME = “NDW050”
        And TABLE_NAME = “APERIOD”
        Set BEGIN_MOD_DATE to maximum MODIFICATION_DATE from APERIOD
        table.

    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW050.SQL”
        DATE_TIME = system date/time
        REFRESH_ID = v.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        MODIFICATION_DATE = maximum MODIFICATION_DATE from APERIOD
        table
        RECORDS_PROCESSED = v.RECORD_COUNT
        TABLE_NAME_PROCESSED = “APERIOD”
        STEP_DESCR = “NDW-000TBD:  Process to update
        NDW_ACCT_PERIOD_STATUS table successfully completed.”
        All other fields remain null.
```

Update NDW_GL_ACCT_SUMMARY Records

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG and terminate processing.

```
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW050.SQL”
        DATE_TIME = system date/time
        REFRESH_ID = v.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Write record to log table.

```
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW050.SQL”
    DATE_TIME = system date/time
    REFRESH_ID = v.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    TABLE_NAME_PROCESSED = “NDW_GL_ACCT_SUMMARY”
    STEP_DESCR = “NDW-000TBD:  Routine to update MONTH_CLOSED_FLAG
started.”
    All other fields remain null.
```

Process NDW_GL_ACCT_SUMMARY records.

```
Read NDW_GL_ACCT_SUMMARY records
    Where MONTH_CLOSED_FLAG = “N”
    And GL_END_DATE = NDW_ACCT_PERIOD_STATUS.GL_END_DATE
    And NAPS.GL_STATUS = “C”.

    Update NDW_GL_ACCT_SUMMARY table:
        Set NGAS.MONTH_CLOSED_FLAG = “Y”.
        Set NGAS. NDW_LAST_MOD_DATE to system date
        Set NGAS.NDW_LAST_MOD_USER_NAME to user’s name
        Set NGAS.NDW_LAST_MOD_DEVICE_NAME to device

    Add 1 to v.RECORD_COUNT
```

When all records have been processed, update the NDW_PROCESS_LOG and commit records.

```
    If last NDW_GL_ACCT_SUMMARY record
        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW050.SQL”
            DATE_TIME = system date/time
            REFRESH_ID = v.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            TABLE_NAME_PROCESSED = “NDW_GL_ACCT_SUMMARY”
            RECORDS_PROCESSED = v.RECORD_COUNT
            STEP_DESCR = “NDW-000TBD:  NDW_GL_ACCT_SUMMARY
            MONTH_CLOSED_FLAG update completed.”.
            All other fields remain null.
        Commit records to database
    End If last
```

After specified number of records have been processed, commit.

```
    After each ND.NDW050_RECORDS_TO_COMMIT records
        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW050.SQL”
            DATE_TIME = system date/time
            REFRESH_ID = v.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            TABLE_NAME_PROCESSED = “NDW_GL_ACCT_SUMMARY”
            RECORDS_PROCESSED = v.RECORD_COUNT
            STEP_DESCR = “NDW-000TBD:  Cumulative records committed to
            NDW_GL_ACCT_SUMMARY table.”.
            All other fields remain null.
        Commit records to database.
    End After Each

End Loop for Updating NDW_GL_ACCT_SUMMARY
```

Update NDW_FIN_CAT_SUMMARY Records

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG and terminate processing.

```
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW050.SQL”
        DATE_TIME = system date/time
        REFRESH_ID = v.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Write record to log table.

```
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW050.SQL”
    DATE_TIME = system date/time
    REFRESH_ID = v.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    TABLE_NAME_PROCESSED = “NDW_FIN_CAT_SUMMARY”
    STEP_DESCR = “NDW-000TBD:  Routine to update MONTH_CLOSED_FLAG
started.”
    All other fields remain null.
```

Process NDW_FIN_CAT_SUMMARY records.

```
Read NDW_FIN_CAT_SUMMARY records
    Where MONTH_CLOSED_FLAG = “N”
    And GL_END_DATE = NDW_ACCT_PERIOD_STATUS.GL_END_DATE
    And NAPS.GL_STATUS = “C”.

    Update NDW_FIN_CAT_SUMMARY table:
```

```
    Set NFCS.MONTH_CLOSED_FLAG = “Y”.
    Set NFCS. NDW_LAST_MOD_DATE to system date
    Set NFCS.NDW_LAST_MOD_USER_NAME to user’s name
    Set NFCS.NDW_LAST_MOD_DEVICE_NAME to device

Add 1 to v.RECORD_COUNT
```

When all records have been processed, update the NDW_PROCESS_LOG and commit records*.*

```
If last NDW_FIN_CAT_SUMMARY record
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW050.SQL”
        DATE_TIME = system date/time
        REFRESH_ID = v.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        TABLE_NAME_PROCESSED = “NDW_FIN_CAT_SUMMARY”
        RECORDS_PROCESSED = v.RECORD_COUNT
        STEP_DESCR = “NDW-000TBD:  NDW_FIN_CAT_SUMMARY
        MONTH_CLOSED_FLAG update completed.”.
        All other fields remain null.
    Commit records to database
End If last
```

After specified number of records have been processed, commit.

```
After each ND.NDW050_RECORDS_TO_COMMIT records
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW050.SQL”
        DATE_TIME = system date/time
        REFRESH_ID = v.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        TABLE_NAME_PROCESSED = “NDW_FIN_CAT_SUMMARY”
        RECORDS_PROCESSED = v.RECORD_COUNT
        STEP_DESCR = “NDW-000TBD:  Cumulative records committed to
        NDW_FIN_CAT_SUMMARY table.”.
        All other fields remain null.
    Commit records to database.
End After Each
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG, commit records and terminate processing.

```
Read VALUE from NDW_DEFAULTS
    Where ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”.
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW050.SQL”
            DATE_TIME = system date/time
            REFRESH_ID = v.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            TABLE_NAME_PROCESSED = “NDW_FIN_CAT_SUMMARY”
            RECORDS_PROCESSED = v.RECORD_COUNT
            STEP_DESCR = “NDW-000TBD:  Process terminated due to
            END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
            All other fields remain null.
    Commit records to database.
    Terminate processing.
    End If END_NDW_BATCH_PROCESSES_FLAG

End Loop for Updating NDW_FIN_CAT_SUMMARY
```

Close Routine

Write all NDW_PROCESS_LOG records written for this routine to the .log file and close file.
Terminate processing.

4.3 NOAA Data Mart Transaction Table Refresh

The NOAA Data Mart transaction tables provide detailed transaction data based on the TRIAL entries, the Budget tables, and detailed labor data from CFS. The transaction level tables are as follows:

Based on TRIAL entries:

- NDW_COMMIT_TRANS [priority 3]
- NDW_AP_TRANS

Based on BUDGET_CONTROL and BUDGET_DETAIL tables:

- NDW_BOP_DETAIL

Based on GJ_CONTROL, GJ_DETAIL, and GJ_EMPLOYEE:

- NDW_LABOR_DETAIL

The following subsections provide the detailed specifications for populating these tables. Appendix B – Transaction Table Record Formats, provides the database table format for these tables.

4.3.1 Refresh Commitment Transaction Tables From Trial (NDW010_REFRESH.SQL)

This is a priority 3 item and will be defined at a later time.

4.3.2 Refresh Accounts Payable Transaction Table From Trial (NDW011_REFRESH.SQL)

The accounts payable transaction table (NDW_AP_TRANS) stores a transaction record for each accounting event recorded within TRIAL that affects standard general ledger accounts between 4800 and 4999 (both the creation and reversal of obligations and expenses).

This routine reads the new TRIAL transactions since the last NOAA Data Mart refresh was run and creates new transaction records for expenses in the NDW_AP_TRANS table based on their standard general ledger account number in TRIAL.

Routine Name

NDW011_REFRESH.SQL

Input:

This routine will be invoked through a regularly scheduled cron job and uses the following data as input:

- CAMS Data Warehouse Tables:

    - TRIAL (TR)
    - CHART (CH)
    - DW_EMPLOYEE_DIM (DED)
    - VENDOR_DETAIL (VD)
    - CUSTOMER (C)
    - CUSTOMER_CONTACT (CC)
    - AP_CONTROL (AC)
    - AP_DETAIL (AD)
    - PO_CONTROL (PC)
    - PO_ITEM (PI)
    - RT_CONTROL (RT)
    - RT_ITEM (RI)
- NOAA Data Mart Tables:
    - NDW_AP_TRANS (NAT)
    - NDW_REFRESH_PARAMS (NRP)
    - NDW_DEFAULTS (ND)
    - NDW_REFRESH_RUN_CONTROL (NRRC)
    - NDW_ACCS_ID_MAP (NAIM)
    - NDW_FIN_CAT_DEF_DETAIL (NFCDD)
    - CBS_TRANSACTION_D (CTD)
    - CBS_TRANS_ITEM_D (CTID)
    - CBS_CARD_HOLDER_L (CCHL)
    - CBS_NOTES_D (CND)

Output:

This routine will produce the following return parameters, output files, database tables, and/or reports:

- NOAA Data Mart Tables:
    - NDW_AP_TRANS (NAT)
    - NDW_REFRESH_PARAMS (NRP)
    - NDW_REFRESH_RUN_CONTROL (NRRC)
    - NDW_PROCESS (NP)
- Reports:
    - NDW011_RE999_PASS999.log (if database is available).
    - NDW011_yyyymmdd.log (if database is not available).
    - Only keep 10 versions of NDW011 log files in the directory.

Processing Logic:

```
Begin Routine
Write date/time started to .log file.
Write header information to .log file.

Determine refresh run.  There should be only one refresh run active at any time, although
multiple processes may be run in parallel.
Read REFRESH_ID_NO, NDW011_REFRESH_PASS_NO,
CFS_SNAPSHOTS_COMPLETE
    From NDW_REFRESH_RUN_CONTROL (NRRC)
    Where ACTIVE_STATUS = “Y”.

If more than one entry
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW011_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  More than one active
        NDW_REFRESH_RUN_CONTROL entry.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW011_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No active NDW_REFRESH_RUN_CONTROL
        entry found.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

Set v.REFRESH_PASS_NO = NDW011_REFRESH_PASS_NO + 1
```

If the snapshots of the CFS tables are not complete, terminate the job.

```
If NRRC.CFS_SNAPSHOTS_COMPLETE = “N”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW011_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Snapshots of CFS tables not complete.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Determine maximum number of transactions to process and how often to commit.

```
Read VALUE(s)
    From NDW_DEFAULTS (ND) for the following ITEM_NAMEs
    ITEM_NAME = “NDW011_RECORDS_TO_PROCESS”
    ITEM_NAME = “NDW011_RECORDS_TO_COMMIT”
    ITEM_NAME = “NDW_AP_TRANS_PRIVACY”
    ITEM_NAME = “NDW_AP_TRANS_TABLE_ID”
    ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”
If any entry cannot be found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW011_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Entry not found in NDW_DEFAULTS for
        ITEM_NAME: ??”.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG and terminate processing.

```
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW011_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Terminate processing.
End If END_NDW_BATCH_PROCESSES_FLAG
```

Determine beginning transaction number that should be used for selecting records from TRIAL for this routine.

```
Read LAST_TRANS_NO_PROCESSED from NDW_REFRESH_PARAMS (NRP)
    Where ROUTINE_NAME = “NDW011_REFRESH”
    And TABLE_NAME = “TRIAL”.
If no entry found
```

```
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW011_REFRESH.SQL”
    REFRESH_ID = NRRC.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    DATE_TIME = system date/time
    STEP_DESCR = “NDW-000TBD:  No entry found for routine in
    NDW_REFRESH_PARAMS table.”
    All other fields remain null.
Write NDW_PROCESS_LOG entry to .log file.
Terminate processing.
```

Write record to log table.

```
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW011_REFRESH.SQL”
    REFRESH_ID = NRRC.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    DATE_TIME = system date/time
    TRANS_NO = NRP.LAST_TRANS_NO_PROCESSED
    TABLE_NAME_PROCESSED = NRP.TABLE_NAME
    STEP_DESCR = “NDW-000TBD:  Routine started with TRIAL_ID > this number.”
    All other fields remain null.
```

Read TRIAL Records

Process only TRIAL records since the last refresh.

```
For each record within TRIAL (TR) where
    TRIAL_ID > NRP.LAST_TRANS_NO_PROCESSED and <
    NRP.LAST_TRANS_NO_PROCESSED + ND.NDW011_RECORDS_TO_PROCESS
```

Insert NDW_AP_TRANS Records

Insert NDW_AP_TRANS table based on TRIAL records

Set NDW_AP_TRANS fields to TRIAL fields as reflected in Appendix B – Transaction Table Record Formats.

Set MONTH_CLOSED_FLAG = “N”.

Set PRIVACY_ACT_APPLIES to ND.PRIVACY_NDW_AP_TRANS.

Update NDW Table Indicator and Trans No.

Set TRANS_TABLE_INDICATOR to ND.TABLE_ID_NDW_AP_TRANS.

Set NDW_TRANS_NO to next sequential number from NDW_MAXSEQNOS
Where TABLE_NAME = “NDW_AP_TRANS”.

Update Financial Category Group Identifier

```
Set FIN_CAT_GROUP = FINC_CAT_ID from NDW_FIN_CAT_DEF_DETAIL
(NFCDD)
    Where TR.ACCOUNT_NO = NFCDD.ACCOUNT_NO
    And TR.SUB_ACCOUNT_NO = NFCDD.SUB_ACCOUNT_NO
    And NFCDD.FIN_CAT_GROUP_FLAG = “Y”
    And NFCDD.ACTIVE_STATUS = “Y”
    And TR.GL_END_DATE between NFCDD.START_DATE and
    NFCDD.END_DATE
```

Update NDW_ACCS_ID

```
Match TR.TRIAL_ID to NAIM.TRIAL_ID
    Set NDW_ACCS_ID to NAIM.NDW_ACCS_ID
If no match found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW011_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  WARNING:  Unique ACCS ID could not
        be found for this TRIAL_ID.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Do not terminate processing.
```

Update flag to indicate this is a prior year record

```
If TR.ACCOUNT_NO = 4880, 4881, 4882, 4980, 4981, 4982
    Set PY_RECORD_FLAG = “Y”
Else
    Set PY_RECORD_FLAG = “N”
```

Derive Feeder System – for batch interfaces, this field indicates the source system of the interface.[2]

```
If TR.SUBSYSTEM_CODE = “AP”
    And TR.TRANS_SOURCE = “AP”
    Use AP_CONTROL.CREATED_BY if = “PURCHASE CARD” or “TRAVEL
    MANAGER”.

If TR.SUBSYSTEM_CODE = “APC”
    And TR.TRANS_SOURCE = “APC”
    Use APC_ACCOUNT.USER_NAME if = “PURCHASE CARD”
```

Derive Employee Name

```
If TR.EMP_NO is not null
    Read DW_EMPLOYEE_DIM (DED)
        Where DED.EMP_NO = TR.EMP_NO
    If a match is found
        Set DAT.EMP_LAST_NAME to DED.LAST_NAME,
        Set DAT.EMP_FIRST_NAME to DED.FIRST_NAME
        Set DAT.EMP_MIDDLE_NAME to DED.MIDDLE_NAME
    Else
        Set DAT.EMP_LAST_NAME = “NOT FOUND”.
```

Derive Vendor ID Name

```
If TR.VENDOR_NO is not null and TR.VENDOR_ID is not null
    Read VENDOR_DETAIL (VD)
        Where VD.VENDOR_NO = TR.VENDOR_NO
        And VD.VENDOR_ID = TR.VENDOR_ID
    If a match is found
        Set DAT.VENDOR_ID_NAME to VD.ADDRESS_NAME
    Else
```

[2] NOTE: This logic may change once the standard interface is implemented for Accounts Payable as it may introduce a feeder system field name in the source tables.

```
        Set DAT.VENDOR_ID_NAME = “NOT FOUND”.
```

Derive Customer Name

```
If TR.CUSTOMER_NO is not null and TR.CUSTOMER_NO is not 0
    Read CUSTOMER (C)
        Where C.CUSTOMER_NO = TR.CUSTOMER_NO
    If a match is found
        Set DAT.CUSTOMER_NAME to C.CUSTOMER_NAME
    Else
        Set DAT.CUSTOMER_NAME = “NOT FOUND”.
```

Derive Customer Point-of-Contact Name

```
If TR.CUSTOMER_NO is not null and TR.CUSTOMER_NO is not 0
    And TR.CONTACT_NO is not null
    Read CUSTOMER_CONTACT (CC)
        Where CC.CUSTOMER_NO = TR.CUSTOMER_NO
        And CC.CONTACT_NO = TR.CONTACT_NO
    If a match is found
        Set DAT.CONTACT_NAME to CC.CONTACT_NAME
    Else
        Set DAT.CONTACT_NAME = “NOT FOUND”.
```

Determine NDW_ACCS_ID

```
Read NDW_ACCS_ID_MAP (NAIM)
    Where NAIM.TRIAL_ID = TR.TRIAL_ID

If no match found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW011_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = TR.TRIAL_ID
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        STEP_DESCR = “NDW-000TBD:  WARNING: Unique ACCS ID could not be
        found for this TRIAL_ID.”
        All other fields remain null.
    Continue processing
Else
    Set NAT.NDW_ACCS_ID = NAIM.NDW_ACCS_ID
```

Derive Document Key Fields

Based on TRIAL ACCOUNT_NO, SUBSYSTEM_CODE, TRANS_SOURCE, the DOCUMENT fields and the ORG_DOCUMENT fields, determine the following (as applicable):

- Document affected by the transaction (fields with “AFFECTED” prefix)
- Obligation document numbers (fields with “PO” prefix)
- Estimated Accrual document numbers (fields with “EA” prefix)
- Receiving Ticket document numbers (fields with “RT” prefix)
- Vendor Invoice document numbers (fields with “INV” prefix)
- Disbursement document numbers (fields with “DISB” prefix)
- Vendor Invoice Correction document numbers (fields with “INVCOR” prefix)
- Voided document numbers (fields with “VOID” prefix)
- General Ledger document numbers (fields with “GJ” prefix)

Derive Commerce Purchase Card System Data

```
If TR.ITEM_TYPE = "PCARD"
    And TR.DOCUMENT_TYPE = "VINV"
    Read AP_DETAIL (AD)
        Where AD.TRANS_NO = TR.DOCUMENT_NO
        And AD.ITEM_NO = TR.ITEM_NO
        And AD.LINE_NO = TR.LINE_NO
Else If
    TR.ORG_ITEM_TYPE = "PCARD"
    And TR.ORG_DOCUMENT_TYPE = "VINV"
    Read AP_DETAIL (AD)
        Where AD.TRANS_NO = TR.ORG_DOCUMENT_NO
        And AD.ITEM_NO = TR.ORG_ITEM_NO
        And AD.LINE_NO = TR.ORG_LINE_NO
Else (not a CPCS transaction)
    Proceed to next edit.

If no match found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW011_REFRESH.SQL"
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = TR.TRIAL_ID
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        STEP_DESCR = "NDW-000TBD:  WARNING: Purchase Card information
        could not be matched to AP table for this TRIAL_ID."
        All other fields remain null.
    Proceed to next edit
Else
    Set NAT.CBS_TRANS_SEQ_ID = positions 3 - 12 of AP.ITEM_DESCR
    Set NAT.CBS_ITEM_NO = positions 14 - 16 of AP.ITEM_DESCR
    Set NAT.CBS_LINE_NO = positions 17 - 18 of AP.ITEM_DESCR

If NAT.CBS_TRANS_SEQ_ID, NAT.CBS_ITEM_NO, or NAT.CBS_LINE_NO are
not positive integers
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW011_REFRESH.SQL"
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = TR.TRIAL_ID
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        STEP_DESCR = "NDW-000TBD:  WARNING: Purchase Card information
        could not be derived from AP_DETAIL ITEM_DESCR."
        All other fields remain null.
    Proceed to next edit
```

Obtain Commerce Purchase Card System item level Data

```
    Read CBS_TRANS_ITEM_D (CTID)
        Where CTID.TRANS_SEQ_ID = NAT.CBS_TRANS_SEQ_ID
        And CTID.ITEM_NO = NAT.ITEM_NO
        If no match found
            Write record to NDW_PROCESS_LOG where
                ROUTINE_NAME = "NDW011_REFRESH.SQL"
                REFRESH_ID = NRRC.REFRESH_ID_NO
                PASS_NO = v.REFRESH_PASS_NO
                DATE_TIME = system date/time
                TRANS_NO = TR.TRIAL_ID
                TABLE_NAME_PROCESSED = NRP.TABLE_NAME
                STEP_DESCR = "NDW-000TBD:  WARNING: Purchase Card
                information could not be matched to CBS_TRANS_ITEM_D for this
                TRIAL_ID."
                All other fields remain null.
            Set NAT.CBS_MERCHANT_NAME to "NOT FOUND".
            Set NAT.REF_NUM to "NOT FOUND".
            Proceed to next edit
        Else
            Set NAT.CBS_PURCH_DATE to CTID.PURCH_DATE
            Set NAT.CBS_MERCHANT_NAME to CTID.MERCHANT_NAME
            Set NAT.REF_NUM to CTID.REF_NUM
```

Obtain Commerce Purchase Card System card holder level Data

```
Read CBS_TRANSACTION_D (CTD) and CBS_CARD_HOLDER_L (CCHL)
    Where CTD.TRANS_SEQ_ID = NAT.CBS_TRANS_SEQ_ID
    And CTD.CARD_NUM = CCHL.CARD_NUMBER
    If no match found
        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = "NDW011_REFRESH.SQL"
            REFRESH_ID = NRRC.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            DATE_TIME = system date/time
            TRANS_NO = TR.TRIAL_ID
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            STEP_DESCR = "NDW-000TBD:  WARNING: Purchase Card
            information could not be matched to CBS_CARD_HOLDER_L for
            this TRIAL_ID."
            All other fields remain null.
        Proceed to next edit
    Else
        Set NAT.CBS_CARD_NUMBER to last 6 positions of
        CCHL.CARD_NUMBER
        Set NAT.CBS_ASC_CODE to CCHL.L3_PARENT_ORG

Read DW_EMPLOYEE_DIM (DED)
    Where DED.EMP_NO = CCHL.EMP_NO
If no match is found
        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = "NDW011_REFRESH.SQL"
            REFRESH_ID = NRRC.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            DATE_TIME = system date/time
            TRANS_NO = TR.TRIAL_ID
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            STEP_DESCR = "NDW-000TBD:  WARNING: Purchase Cardholder
            name could not be matched to DW_EMPLOYEE_DIM for this
            TRIAL_ID."
            All other fields remain null.
        Set DAT.CBS_CH_LAST_NAME to "NOT FOUND".
        Proceed to next edit
Else
    Set DAT.CBS_CH_LAST_NAME to DED.LAST_NAME.
    Set DAT.CBS_CH_FIRST_NAME to DED.FIRST_NAME.
    Set DAT.CBS_CH_MIDDL_NAME to DED.MIDDLE_NAME.
```

Obtain Commerce Purchase Card System item level notes data

```
    Read first CBS_NOTES_D (CND)
        Where CND.TRANS_SEQ_ID = NAT.CBS_TRANS_SEQ_ID
            And CND.ITEM_NO = NAT.CBS_ITEM_NO
        If no match found
            Write record to NDW_PROCESS_LOG where
                ROUTINE_NAME = “NDW011_REFRESH.SQL”
                REFRESH_ID = NRRC.REFRESH_ID_NO
                PASS_NO = v.REFRESH_PASS_NO
                DATE_TIME = system date/time
                TRANS_NO = TR.TRIAL_ID
                TABLE_NAME_PROCESSED = NRP.TABLE_NAME
                STEP_DESCR = “NDW-000TBD:  WARNING: Purchase Card
                information could not be matched to CBS_NOTES_D for this
                TRIAL_ID.”
                All other fields remain null.
            Set NAT.CBS_NOTES to “NOT FOUND”
            Proceed to next edit
        Else
            Set NAT.CBS_NOTES to CND.NOTE
```

Derive FISCAL_MONTH field from GL_END_DATE

```
If TR.GL_END_DATE month > 9
    Set NAT.FISCAL_MONTH = TR.GL_END_DATE month - 9
Else
    Set NAT.FISCAL_MONTH = TR.GL_END_DATE month + 3
```

Set BALANCE_FLAG based on ACCOUNT_NO/SUB_ACCOUNT_NO in CHART

```
Set NAT.BALANCE_FLAG to CH.BALANCE_FLAG from CHART (CH)
    Where CH.ACCOUNT_NO = TR.ACCOUNT_NO
    And CH.SUB_ACCOUNT_NO = TR.SUB_ACCOUNT_NO
```

Update flag to indicate this is the reversing entry to a prior transaction

```
If (NAT.BALANCE_FLAG = “DR” and TR.CREDIT_AMOUNT is not 0 or null) or
    (NAT.BALANCE_FLAG = “CR” and TR.DEBIT_AMOUNT is not 0 or null)
    Set REVERSE_ENTRY_FLAG = “Y”
Else
    Set  REVERSE_ENTRY_FLAG = “N”
```

Compute Net Amount based on BALANCE_FLAG for the SGL account.

```
If NAT.BALANCE_FLAG = “DR”
    Set NAT.NET_AMOUNT = TR.DEBIT_AMOUNT – TR.CREDIT_AMOUNT
If NAT.BALANCE_FLAG = “CR”
    Set NAT.NET_AMOUNT = TR.CREDIT_AMOUNT – TR.DEBIT_AMOUNT
If NAT.BALANCE_FLAG = “DC”
    Set NAT.NET_AMOUNT to 0.
```

Summarized Labor Record Processing
Update flag to indicate this is a labor record and add STAT_UNIT_QTY

```
If TR.ACCOUNT_NO between 4900 and 4999
```

```
    And TR.SUBSYSTEM_CODE = “GJ”
    And TR.TRANS_SOURCE = “GJ”
    And substr(TR.TRANS_DESCR,13,6) in (“NFC002”,”NFC005”,”NFC004”)
        Set NAT.LABOR_HOURS to TR.STAT_UNIT_QTY
        Set LABOR_RECORD_FLAG = “Y”
Else
        Set LABOR_RECORD_FLAG = “N”
```

Indicate type of labor record type

```
If TR.ACCOUNT_NO between 4900 and 4999
    And TR.SUBSYSTEM_CODE = “GJ”
    And TR.TRANS_SOURCE = “GJ”
    If substr(TR.TRANS_DESCR,13,6) = “NFC002”
        Set NAT.LABOR_HOURS to TR.STAT_UNIT_QTY
        Set LABOR_RECORD_FLAG = “Y”
        Set NDW_LABOR_TYPE = “NFC” (actual labor from NFC interface)
    Else if substr(TR.TRANS_DESCR,13,6) = “NFC005”
        Set NAT.LABOR_HOURS to TR.STAT_UNIT_QTY
        Set LABOR_RECORD_FLAG = “Y”
        Set NDW_LABOR_TYPE = “ADJ” (labor adjustment thru NFC005 screen)
    Else if substr(TR.TRANS_DESCR,13,6) = “NFC004”
        Set NAT.LABOR_HOURS to TR.STAT_UNIT_QTY
        Set LABOR_RECORD_FLAG = “Y”
        Set NDW_LABOR_TYPE = “DEF” (labor default applied thru NFC004)
    Else link record to GJ_CONTROL
    If GC.REF like “EST%”
        If GC.DESC like “NFC008%”
            If GC.REVERSE_FLAG = “Y”
                Set NAT.LABOR_HOURS to TR.STAT_UNIT_QTY
                Set LABOR_RECORD_FLAG = “Y”
                Set NDW_LABOR_TYPE = “EST” (month end labor estimate)
            If GC.REVERSE_FLAG =  “N”
                Set NAT.LABOR_HOURS to TR.STAT_UNIT_QTY
                Set LABOR_RECORD_FLAG = “Y”
                Set NDW_LABOR_TYPE = “ESTR” (reversal of month end labor
                estimate)
Else
    Set LABOR_RECORD_FLAG = “N”
```

Populate both CREATION and LAST_MOD fields.

```
Set NAT.NDW_CREATION_DATE to system date
Set NAT.NDW_CREATION_USER_NAME to user’s name
Set NAT.NDW_CREATION_DEVICE_NAME to device

Set NAT.NDW_LAST_MOD_DATE to system date
Set NAT.NDW_LAST_MOD_USER_NAME to user’s name
Set NAT.NDW_LAST_MOD_DEVICE_NAME to device

Set all Line Office extract fields to null
```

Update control counts.

```
    Add 1 to v.RECORD_COUNT.
    Add (TR.DEBIT_AMOUNT – TR.CREDIT_AMOUNT) to v.AMOUNT_PROCESSED.
```

When all records have been processed, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG and commit records.

```
If last TRIAL record
    Update NDW_REFRESH_PARAMS
        Where ROUTINE_NAME = “NDW011_REFRESH”
        Set LAST_TRANS_NO_PROCESSED = TR.TRIAL_ID
        Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
        Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

    Update NDW_REFRESH_RUN_CONTROL
        Set NDW011_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
        Set NDW011_MAX_VALUE_PROCESSED = TR.TRIAL_ID

    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW011_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = last TRIAL_ID read from TRIAL
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        AMOUNT_PROCESSED = .AMOUNT_PROCESSED
        STEP_DESCR = “NDW-000TBD:  Process completed.  Last TRIAL_ID
        processed updated”.
        All other fields remain null.
    Commit records to database
End If last
```

After specified number of records have been processed, commit.

```
After each  ND.NDW011_RECORDS_TO_COMMIT records
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW011_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = TR.TRIAL_ID
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
        STEP_DESCR = “NDW-000TBD:  Cumulative records committed to
        NDW_AP_TRANS table.”.
        All other fields remain null.
    Commit records to database.
End After Each
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG, commit records and terminate processing.

```
Read VALUE from NDW_DEFAULTS
    Where ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”.
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Update NDW_REFRESH_PARAMS
        Where ROUTINE_NAME = “NDW011_REFRESH”
        Set LAST_TRANS_NO_PROCESSED = TR.TRIAL_ID
        Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
```

```
            Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

        Update NDW_REFRESH_RUN_CONTROL
            Set NDW011_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
            Set NDW011_MAX_VALUE_PROCESSED = TR.TRIAL_ID

        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW011_REFRESH.SQL”
            REFRESH_ID = NRRC.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            DATE_TIME = system date/time
            TRANS_NO = last TRIAL_ID read from TRIAL
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            RECORDS_PROCESSED = v.RECORD_COUNT
            AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
            STEP_DESCR = “NDW-000TBD:  Process terminated due to
            END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
            All other fields remain null.
        Commit records to database.
        Terminate processing.
    End If END_NDW_BATCH_PROCESSES_FLAG
End If TRANS_SOURCE <> “BEGBAL”
End Loop for Matching to NDW_AP_TRANS
```

Close Routine

```
Write all NDW_PROCESS_LOG records written for this routine to the .log file and close
file.
If ND.END_NDW_BATCH_PROCESSED_FLAG = “N”
    And NRRC.NDW011_MAX_VALUE_PROCESSED < NRRC.MAX_TRIAL
    Restart the routine at the beginning for an additional pass until all records are processed.
Else
    Terminate processing.
```

4.3.3 Refresh Transaction Tables From Budget Tables (NDW012_REFRESH.SQL)

The purpose of this table is to provide detailed Budget Operating Plan data for Line/Field Office extract and reporting purposes.

Routine Name

NDW012_REFRESH.SQL

Input:

This routine will be invoked through a regularly scheduled cron job and uses the following data as input:

- CAMS Data Warehouse Tables:
    - BUDGET_CONTROL (BC)
    - BUDGET_DETAIL (BD)
- NOAA Data Mart Tables:
    - NDW_BOP_DETAIL (NBD)
    - NDW_REFRESH_PARAMS (NRP)
    - NDW_DEFAULTS (ND)
    - NDW_REFRESH_RUN_CONTROL (NRRC)

Output:

This routine will produce the following return parameters, output files, database tables, and/or reports:

- NOAA Data Mart Tables:
    - NDW_BOP_DETAIL (NDS)
    - NDW_REFRESH_PARAMS (NRP)
    - NDW_REFRESH_RUN_CONTROL (NRRC)
    - NDW_PROCESS (NP)
- Reports:
    - NDW012_RE999_PASS999.log (if database is available).
    - NDW012_yyyymmdd.log (if database is not available).

- Only keep 10 versions of NDW012 log files in the directory.

Processing Logic:

Begin Routine

Write date/time started to .log file.

```
Write header information to .log file.
```

Determine refresh run. There should be only one refresh run active at any time, although multiple processes may be run in parallel.

```
Read REFRESH_ID_NO, NDW012_REFRESH_PASS_NO,
CFS_SNAPSHOTS_COMPLETE
    From NDW_REFRESH_RUN_CONTROL (NRRC)
    Where ACTIVE_STATUS = "Y".

If more than one entry
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW012_REFRESH.SQL"
        DATE_TIME = system date/time
        STEP_DESCR = "NDW-000TBD:  More than one active
        NDW_REFRESH_RUN_CONTROL entry."
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW012_REFRESH.SQL"
        DATE_TIME = system date/time
        STEP_DESCR = "NDW-000TBD:  No active NDW_REFRESH_RUN_CONTROL
        entry found."
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

Set v.REFRESH_PASS_NO = NDW012_REFRESH_PASS_NO + 1
```

If the snapshots of the CFS tables are not complete, terminate the job.

```
If NRRC.CFS_SNAPSHOTS_COMPLETE = "N"
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW012_REFRESH.SQL"
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = "NDW-000TBD:  Snapshots of CFS tables not complete."
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Determine maximum number of transactions to process and how often to commit.

```
Read VALUE(s)
    From NDW_DEFAULTS (ND) for the following ITEM_NAMEs
    ITEM_NAME = “NDW012_RECORDS_TO_COMMIT”
    ITEM_NAME = “NDW_BOP_DETAIL_PRIVACY”
    ITEM_NAME = “NDW_BOP_DETAIL_TABLE_ID”
    ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”
If any entry cannot be found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW012_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Entry not found in NDW_DEFAULTS for
        ITEM_NAME: ??”.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG and terminate processing.

```
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW012_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
End END_NDW_BATCH_PROCESSES_FLAG
```

Determine last modification date and trans_no fields that should be used for selecting records from BUDGET_CONTROL and BUDGET_DETAIL for this routine.

```
Read LAST_TRANS_NO_PROCESSED, BEGIN_MOD_DATE
    From NDW_REFRESH_PARAMS (NRP)
    Where ROUTINE_NAME = “NDW012_REFRESH”
    And TABLE_NAME = “BUDGET_CONTROL”.
If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW012_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No entry found for routine in
        NDW_REFRESH_PARAMS table.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Write record to log table.

```
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW012_REFRESH.SQL”
    REFRESH_ID = NRRC.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    DATE_TIME = system date/time
    TRANS_NO = NRP.LAST_TRANS_NO_PROCESSSED
    MODIFICATION_DATE = NRP.BEGIN_MOD_DATE
```

TABLE_NAME_PROCESSED = NRP.TABLE_NAME
STEP_DESCR = “NDW-000TBD: Routine started with key fields > this date and number.”
All other fields remain null.
Write NDW_PROCESS_LOG entry to .log file.
Terminate processing.

Read BUDGET_CONTROL & BUDGET_DETAIL Records
Process only BUDGET_CONTROL/DETAIL records since the last refresh.

Note that because budget records can be approved out of sequence and there is no unique identifier for the BUDGET_DETAIL record, there is no RECORDS_TO_PROCESS field associated with these records. All newly approved BUDGET_CONTROL and BUDGET_DETAIL records are selected for processing.

Read BUDGET_CONTROL & BUDGET_DETAIL records
Where BC.APPROVED_FLAG = “Y”
And BC.MODIFICATION_DATE ||[3] BC.TRANS_NO > NRP.BEGIN_MOD_DATE || NRP.LAST_TRANS_NO_PROCESSED

Order by BC.MODIFICATION_DATE, BC.TRANS_NO

Insert NDW_BOP_DETAIL Records
Insert NDW_BOP_DETAIL record

Create a new record in the NDW_BOP_DETAIL table for each BUDGET_DETAIL record processed.

Set NDW_BOP_DETAIL fields to BUDGET_CONTROL/BUDGET_DETAIL fields as reflected in Appendix B – Transaction Table Record Formats.

Set TRANS_TABLE_INDICATOR to ND.TABLE_ID_NDW_BOP_DETAIL.

Set NDW_TRANS_NO to next sequential number from NDW_MAXSEQNOS
Where TABLE_NAME = “NDW_BOP_DETAIL”

Set MONTH_CLOSED_FLAG = “N”.

Set PRIVACY_ACT_APPLIES to ND.PRIVACY_NDW_BOP_DETAIL.

Populate both CREATION and LAST_MOD fields.
Set NDW_CREATION_DATE to system date
Set NDW_CREATION_USER_NAME to user’s name
Set NDW_CREATION_DEVICE_NAME to device

Set NDW_LAST_MOD_DATE to system date
Set NDW_LAST_MOD_USER_NAME to user’s name
Set NDW_LAST_MOD_DEVICE_NAME to device

Set all Line Office extract fields to null

Update control counts.

[3] The symbol || indicates that the fields should be concatenated (strung together).

```
Add 1 to v.RECORD_COUNT.
Add BD.AMOUNT to v.AMOUNT_PROCESSED.
```

When all records have been processed, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG and commit records.

```
If last BUDGET_DETAIL record
    Update NDW_REFRESH_PARAMS
        Where ROUTINE_NAME = “NDW012_REFRESH”
        Set LAST_TRANS_NO_PROCESSED = BC.TRANS_NO
        Set BEGIN_MOD_DATE = BC.MODIFICATION_DATE
        Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
        Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

    Update NDW_REFRESH_RUN_CONTROL
        Set NDW012_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
        Set NDW012_MAX_VALUE_PROCESSED = BC.TRANS_NO
        Set NDW012_LAST_MOD_DATE = BC.MODIFICATION_DATE

    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW012_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = BC.TRANS_NO
        MODIFICATION_DATE = BC.MODIFICATION_DATE
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
        STEP_DESCR = “NDW-000TBD:  Process completed.  Last approved
        BUDGET_CONTROL record with this key processed”.
        All other fields remain null.
    Commit records to database
End If last
```

After specified number of records have been processed, commit.

```
After each ND.NDW012_RECORDS_TO_COMMIT records

    Check to see if the next BUDGET_DETAIL record has the same TRANS_NO as the
    last one processed.
```

Because the BUDGET_DETAIL record does not have a unique key, the BUDGET_DETAIL records must be processed until the BUDGET_CONTROL TRANS_NO key changes.

```
    If so, continue processing the records until the next record’s TRANS_NO changes.

    Else
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW012_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = BC.TRANS_NO
        MODIFICATION_DATE = BC.MODIFICATION_DATE
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
```

```
            RECORDS_PROCESSED = v.RECORD_COUNT
            AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
            STEP_DESCR = “NDW-000TBD:  Cumulative records committed to
            NDW_BOP_DETAIL table.”.
            All other fields remain null.
        Commit records to database.
    End After Each
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG, commit records and terminate processing.

```
    Read VALUE from NDW_DEFAULTS
        Where ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”.
    If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
        Update NDW_REFRESH_PARAMS
            Where ROUTINE_NAME = “NDW012_REFRESH”
            Set LAST_TRANS_NO_PROCESSED = BC.TRANS_NO
            Set BEGIN_MOD_DATE = BC.MODIFICATION_DATE
            Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
            Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

        Update NDW_REFRESH_RUN_CONTROL
            Set NDW012_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
            Set NDW012_MAX_VALUE_PROCESSED = BC.TRANS_NO
            Set NDW012_LAST_MOD_DATE = BC.MODIFICATION_DATE

        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW012_REFRESH.SQL”
            REFRESH_ID = NRRC.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            DATE_TIME = system date/time
            TRANS_NO = BC.TRANS_NO
            MODIFICATION_DATE = BC.MODIFICATION_DATE
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            RECORDS_PROCESSED = v.RECORD_COUNT
            AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
            STEP_DESCR = “NDW-000TBD:  Process terminated due to
            END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
            All other fields remain null.
        Commit records to database.
        Terminate processing.

    End If END_NDW_BATCH_PROCESSES_FLAG
End Loop for Matching to NDW_BOP_DETAIL
```

Close Routine

Write all NDW_PROCESS_LOG records written for this routine to the .log file and close file.

Terminate processing.

4.3.4 Refresh Detailed Labor Data (NDW013_REFRESH.SQL)

The labor detail table (NDW_LABOR_DETAIL) stores the detailed labor records as received by the National Finance Center (NFC) through the NFC002 interface, labor detail adjustments made through the CFS NFC005 screen, and month end transaction defaults generated through the NFC004 routine.

This routine reads the new GJ_CONTROL, GJ_DETAIL, and GJ_EMPLOYEE records approved since the last NOAA Data Mart refresh was run and creates new records in the NDW_LABOR_DETAIL table.

Routine Name

NDW013_REFRESH.SQL

Input:

This routine will be invoked through a regularly scheduled cron job and uses the following data as input:

- CAMS Data Warehouse Tables:
    - GJ_CONTROL (GC)
    - GJ_DETAIL (GD)
    - GJ_EMPLOYEE (GE)
    - NFC_UTILITY_VALUES (NUV)
    - EMPLOYEE_CONTROL (EC)
- NOAA Data Mart Tables:
    - NDW_LABOR_DETAIL (NLD)
    - NDW_REFRESH_PARAMS (NRP)
    - NDW_DEFAULTS (ND)
    - NDW_REFRESH_RUN_CONTROL (NRRC)
    - NDW_ACCS_ID_CONTROL (NAIC)

Output:

This routine will produce the following return parameters, output files, database tables, and/or reports:

- NOAA Data Mart Tables:
  - NDW_LABOR_DETAIL (NLD)
  - NDW_REFRESH_PARAMS (NRP)
  - NDW_REFRESH_RUN_CONTROL (NRRC)
  - NDW_PROCESS (NP)
- Reports:
  - NDW013_RE999_PASS999.log (if database is available).
  - NDW013_yyyymmdd.log (if database is not available).
  - Only keep 10 versions of NDW013 log files in the directory.

Processing Logic:

```
Begin Routine
Write date/time started to .log file.
Write header information to .log file.

Determine refresh run.  There should be only one refresh run active at any time, although
multiple processes may be run in parallel.
Read REFRESH_ID_NO, NDW013_REFRESH_PASS_NO,
CFS_SNAPSHOTS_COMPLETE
    From NDW_REFRESH_RUN_CONTROL (NRRC)
    Where ACTIVE_STATUS = “Y”.

If more than one entry
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW013_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  More than one active
        NDW_REFRESH_RUN_CONTROL entry.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW013_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No active NDW_REFRESH_RUN_CONTROL
        entry found.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.

Set v.REFRESH_PASS_NO = NDW013_REFRESH_PASS_NO + 1
```

If the snapshots of the CFS tables are not complete, terminate the job.

```
If NRRC.CFS_SNAPSHOTS_COMPLETE = “N”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW013_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Snapshots of CFS tables not complete.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Determine maximum number of transactions to process and how often to commit.

```
Read VALUE(s)
    From NDW_DEFAULTS (ND) for the following ITEM_NAMEs
    ITEM_NAME = “NDW013_RECORDS_TO_COMMIT”
    ITEM_NAME = “NDW_LABOR_DETAIL_PRIVACY”
    ITEM_NAME = “NDW_LABOR_DETAIL_TABLE_ID”
    ITEM_NAME = “END_NDW_BATCH_PROCESSES_FLAG”
If any entry cannot be found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW013_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Entry not found in NDW_DEFAULTS for
        ITEM_NAME: ??”.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”, update NDW_PROCESS_LOG and terminate processing.

```
If ND.END_NDW_BATCH_PROCESSES_FLAG = “Y”
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW013_REFRESH.SQL”
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  Process terminated due to
        END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
        All other fields remain null.
    Terminate processing.
End If END_NDW_BATCH_PROCESSES_FLAG
```

Determine last modification date and transaction number fields that should be used for selecting records from GJ_CONTROL, GJ_DETAIL, and GJ_EMPLOYEE for this routine.

```
Read LAST_TRANS_NO_PROCESSED and BEGIN_MOD_DATE from
NDW_REFRESH_PARAMS (NRP)
    Where ROUTINE_NAME = “NDW013_REFRESH”
    And TABLE_NAME = “GJ_CONTROL”.
If no entry found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW013_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        STEP_DESCR = “NDW-000TBD:  No entry found for routine in
        NDW_REFRESH_PARAMS table.”
        All other fields remain null.
    Write NDW_PROCESS_LOG entry to .log file.
    Terminate processing.
```

Write record to log table.

```
Write record to NDW_PROCESS_LOG where
    ROUTINE_NAME = “NDW013_REFRESH.SQL”
    REFRESH_ID = NRRC.REFRESH_ID_NO
    PASS_NO = v.REFRESH_PASS_NO
    DATE_TIME = system date/time
    TRANS_NO = NRP.LAST_TRANS_NO_PROCESSED
    TABLE_NAME_PROCESSED = NRP.TABLE_NAME
    STEP_DESCR = “NDW-000TBD:  Routine started with key fields > this date and
    number.”
    All other fields remain null.
```

Read GJ_CONTROL Records

Process only GJ_CONTROL, GJ_DETAIL, and GEJ_EMPLOYEE records since the last refresh.

```
Select GJ_CONTROL & subordinate GJ_DETAIL and GJ_EMPLOYEE records
    Where GC.MANAGER_FLAG = ‘Y’
    And GC.REF begins with ‘NFC’ or ‘NOA’
    And GC.MODIFICATION_DATE || GC.TRANS_NO >
        NRP.BEGIN_MOD_DATE || NRP.LAST_TRANS_NO_PROCESSED

Order by GC.MODIFICATION_DATE, GC.TRANS_NO

For each GJ_EMPLOYEE (GE) record
```

Insert NDW_LABOR_DETAIL Records

Insert NDW_LABOR_DETAIL record based on GJ_EMPLOYEE records

```
    Set NDW_LABOR_DETAIL fields to GJ_CONTROL, GJ_DETAIL, and
    GJ_EMPLOYEE fields as reflected in Appendix B – Transaction Table Record Formats.
```

Update NDW Table Indicator and Trans No.

```
    Set TRANS_TABLE_INDICATOR to ND.TABLE_ID_NDW_LABOR_DETAIL.

    Set NDW_TRANS_NO to next sequential number from NDW_MAXSEQNOS
    Where TABLE_NAME = “NDW_LABOR_DETAIL”.

    Set PRIVACY_ACT_APPLIES to ND.PRIVACY_NDW_LABOR_DETAIL.
```

Update Employee Information

```
    Select EMPLOYEE_CONTROL (EC) record
        Where EC.EMP_NO = GE.EMP_NO

    Set NLD.SSN to EC.SSN
    Set NLD.EMP_FIRST_NAME to EC.FIRST_NAME
    Set NLD.EMP_MIDDLE_NAME to EC.MIDDLE_NAME
    Set NLD.EMP_LAST_NAME to EC.LAST_NAME
```

```
Set NLD.EMP_TITLE to EC.EMP_TITLE
```

Determine NDW_ACCS_ID

```
Read NDW_ACCS_ID_CONTROL (NAIC)
    Matching ACCS fields from GJ records to NAIC ACCS fields.

If no match found
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = “NDW013_REFRESH.SQL”
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = GC.TRANS_NO
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        STEP_DESCR = “NDW-000TBD:  WARNING: Unique ACCS ID could not be
        found for this ACCS.”
        All other fields remain null.
    Continue processing
Else
    Set NAT.NDW_ACCS_ID = NAIC.NDW_ACCS_ID
```

Derive FISCAL_MONTH field from GL_END_DATE

```
If GC.GL_END_DATE month > 9
    Set NAT.FISCAL_MONTH = GC.GL_END_DATE month - 9
Else
    Set NAT.FISCAL_MONTH = GC.GL_END_DATE month + 3
```

Indicate type of labor record type

```
If GC.REF like “NFC%” or “NOA%”
    And GC.DESCR = “NFC002”
        Set NDW_LABOR_TYPE = “NFC” (actual labor from NFC interface)
Else if … logic TBD
        Set NDW_LABOR_TYPE = “ADJ” (labor adjustment thru NFC005 screen)
Else if … logic TBD
        Set NDW_LABOR_TYPE = “DEF” (labor default applied thru NFC004)
Else
    Set NDW_LABOR_TYPE = ‘UNK” (unknown)
```

Populate both CREATION and LAST_MOD fields.

```
Set NLD.NDW_CREATION_DATE to system date
Set NLD.NDW_CREATION_USER_NAME to user’s name
Set NLD.NDW_CREATION_DEVICE_NAME to device

Set NLD.NDW_LAST_MOD_DATE to system date
Set NLD.NDW_LAST_MOD_USER_NAME to user’s name
Set NLD.NDW_LAST_MOD_DEVICE_NAME to device

Set all Line Office extract fields to null
```

Update control counts.

```
Add 1 to v.RECORD_COUNT.
Add (NLD.AMOUNT) to v.AMOUNT_PROCESSED.
```

When all records have been processed, update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG and commit records.

```
If last GJ_EMPLOYEE record
    Update NDW_REFRESH_PARAMS
        Where ROUTINE_NAME = "NDW013_REFRESH"
        Set LAST_TRANS_NO_PROCESSED = GC.TRANS_NO
        Set BEGIN_MOD_DATE = GC.MODIFICATION_DATE
        Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
        Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

    Update NDW_REFRESH_RUN_CONTROL
        Set NDW013_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
        Set NDW013_MAX_VALUE_PROCESSED = GC.TRANS_NO
        Set NDW013_LAST_MOD_DATE = GC.MODIFICATION_DATE

    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW013_REFRESH.SQL"
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = last TRANS_NO read from GJ_CONTROL
        MODIFICATION_DATE = GC.MODIFICATION_DATE
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        AMOUNT_PROCESSED = .AMOUNT_PROCESSED
        STEP_DESCR = "NDW-000TBD:  Process completed.  Last TRANS_NO
        processed updated".
        All other fields remain null.
    Commit records to database
End If last
```

After specified number of records have been processed, commit.

```
After each  ND.NDW013_RECORDS_TO_COMMIT records
    Write record to NDW_PROCESS_LOG where
        ROUTINE_NAME = "NDW013_REFRESH.SQL"
        REFRESH_ID = NRRC.REFRESH_ID_NO
        PASS_NO = v.REFRESH_PASS_NO
        DATE_TIME = system date/time
        TRANS_NO = GC.TRANS_NO
        MODIFICATION_DATE = GC.MODIFICATION_DATE
        TABLE_NAME_PROCESSED = NRP.TABLE_NAME
        RECORDS_PROCESSED = v.RECORD_COUNT
        AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
        STEP_DESCR = "NDW-000TBD:  Cumulative records committed to
        NDW_LABOR_DETAIL table.".
        All other fields remain null.
    Commit records to database.
End After Each
```

If the operator has updated the ND.END_NDW_BATCH_PROCESSES_FLAG = "Y", update the NDW_REFRESH_PARAMS, NDW_REFRESH_RUN_CONTROL, NDW_PROCESS_LOG, commit records and terminate processing.

```
Read VALUE from NDW_DEFAULTS
    Where ITEM_NAME = "END_NDW_BATCH_PROCESSES_FLAG".
If ND.END_NDW_BATCH_PROCESSES_FLAG = "Y"
    Update NDW_REFRESH_PARAMS
        Where ROUTINE_NAME = "NDW013_REFRESH"
```

```
            Set LAST_TRANS_NO_PROCESSED = GC.TRANS_NO
            Set BEGIN_MOD_DATE = GC.MODIFICATION_DATE
            Set NDW_LAST_REFRESH_ID_NO = NRRC.REFRESH_ID_NO
            Set NDW_LAST_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO

        Update NDW_REFRESH_RUN_CONTROL
            Set NDW013_REFRESH_PASS_NO = v.LAST_REFRESH_PASS_NO
            Set NDW013_MAX_VALUE_PROCESSED = GC.TRANS_NO
            Set NDW013_LAST_MOD_DATE = GC.MODIFICATION_DATE

        Write record to NDW_PROCESS_LOG where
            ROUTINE_NAME = “NDW013_REFRESH.SQL”
            REFRESH_ID = NRRC.REFRESH_ID_NO
            PASS_NO = v.REFRESH_PASS_NO
            DATE_TIME = system date/time
            TRANS_NO = last TRANS_NO read from GJ_CONTROL
            MODIFICATION_DATE = GC.MODIFICATION_DATE
            TABLE_NAME_PROCESSED = NRP.TABLE_NAME
            RECORDS_PROCESSED = v.RECORD_COUNT
            AMOUNT_PROCESSED = v.AMOUNT_PROCESSED
            STEP_DESCR = “NDW-000TBD:  Process terminated due to
            END_NDW_BATCH_PROCESSES_FLAG = “Y”“.
            All other fields remain null.
        Commit records to database.
        Terminate processing.
    End If END_NDW_BATCH_PROCESSES_FLAG
End Loop for Matching to NDW_LABOR_DETAIL
```

Close Routine

Write all NDW_PROCESS_LOG records written for this routine to the .log file and close file.

Terminate processing.

4.4 Line/Field Office Support Packages

4.4.1 NDW_RECORD_LO_EXTRACT

In order to ensure consistency in how records are tagged as being extracted, the following database package has been developed for use by the Line/Field Office extract routines. The Line/Field Office extract routine should call this package for each record that is extracted. The routine will populate the Original extract fields if the *pre*_ORIGINAL_EXTRACT_DATE is null and the *pre*_LAST_MOD extract fields for the extracted record in all cases.

Routine Name

NDW_RECORD_LO_EXTRACT

Input:

This routine will be invoked through the various Line/Field Office extract routines:

- Parameters
    - Table Name
    - NDW_TRANS_NO
    - Extract Prefix
    - Calling routine name
- NOAA Data Mart Tables:
    - As identified by Table Name parameter

Output:

This routine will produce the following return parameters, output files, database tables, and/or reports:

- Parameters
    - Table Name
    - NDW_TRANS_NO
    - Extract Prefix
    - Calling routine name
    - Return Code

    - Return Message

- NOAA Data Mart Tables:
    - As identified by Table Name parameter

Processing Logic:

Accept parameters passed from Line/Field Office extract routine.

```
Accept p.TABLE_NAME, p.NDW_TRANS_NO, p.EXTRACT_PREFIX,
v.CALLING_ROUTINE parameters.
```

Update extract fields.

```
Update p.TABLE_NAME (T)
    Where T.NDW_TRANS_NO = p.NDW_TRANS_NO
        Set T.“p.EXTRACT_PREFIX”_LAST_EXTRACT_DATE to system date
        Set T.“p.EXTRACT_PREFIX”_LAST_MOD_USER_NAME to user’s Oracle ID.
        Set T.“p.EXTRACT_PREFIX”_LAST_MOD_DEVICE_NAME to “BATCH “ ||
        p.CALLING_ROUTINE
        Add 1 to “p.EXTRACT_PREFIX”_EXTRACT_ITERATION

        If “p.EXTRACT_PREFIX”_ORIGINAL_EXTRACT_DATE is null
            Set “p.EXTRACT_PREFIX”_ORIGINAL_EXTRACT_DATE to system_date
            Set “p.EXTRACT_PREFIX”_ORIGINAL USER_NAME to user’s Oracle ID.
            Set “p.EXTRACT_PREFIX”_ORIGINAL_DEVICE_NAME to “BATCH “ ||
            p.CALLING_ROUTINE.
        End If

End Update

If record was updated properly, send back p.RETURN_CODE = 0.

If record could not be updated, send back:
    Set p.RETURN_CODE from NDW_RETURN_CODE_CONTROL table based on error
    encountered.

    Set p.RETURN_MESSAGE to SHORT_MESSAGE from
    NDW_RETURN_CODE_CONTROL table based on error encountered.

End Routine
```

4.5 Data Integrity

Data integrity checks will be implemented as part of the NOAA Data Mart. These checks will help ensure that all tables are updated properly and reflect the same base values from the source tables (e.g., TRIAL). Any discrepancies in balances will be highlighted in a report so that appropriate action may be taken.

The NOAA Data Mart integrity checks will be done using only NOAA Data Mart tables. Additional data integrity checks are being implemented by the CAMS Data Warehouse team to ensure that data from production CFS is being accurately transferred to the data warehouse.

4.5.1 Refresh Status Report

At the end of the NOAA Data Mart refresh routine, a series of queries will be performed and a formatted Refresh Status Report produced to indicate the end state of the data. These queries will generate control totals (at a minimum by BUREAU_CODE and FISCAL_YEAR) for each of the tables populated by the NOAA Data Mart refresh routines as well as the source tables from which they are being populated.

A report will then be developed to list all of the control totals obtained and highlight any discrepancies between the tables.

4.5.2 Refresh Run Control Table

The NDW_REFRESH_RUN_CONTROL table will be used by the NDW application administrator to ensure data integrity for a specific refresh run. This table reflects in one row the status of all routines of the refresh process. It indicates:

- The maximum target key fields that should be processed.
- The actual maximum key field processed for each routine.
- The refresh pass number for the routine.
- The status of the refresh.
- If the refresh was stopped by the operator.

By reviewing this record, the operator can determine whether all routines met their target update figures or if there may be a problem with the refresh process.

4.5.3 Process Log

The NDW_PROCESS_LOG is used by every NOAA Data Mart batch routine to record when the routine started, when it completed, and information at key checkpoints within the process.

This table helps ensure data integrity by providing the system administrator the capability to monitor job progress, to determine at what point a routine was terminated, and the manner in which it was terminated.

Appendix A – Summary Table Record Formats

This appendix is provided as separate document called “Appendix A – NOAA Data Mart Summary Table Definitions”.

Appendix B – Transaction Table Record Formats

This appendix is provided as separate document called “Appendix B – NOAA Data Mart Transaction Table Definitions”.

Appendix C – Support Table Record Layouts

This appendix is provided as separate document called "Appendix C – NOAA Data Mart Support Table Definitions".

Appendix D – Reference Table Record Layouts

The following pages contain descriptions of tables currently available within the CAMS Data Warehouse that may be used for reference by users. Many of these tables contain a code followed by a textual description. Other tables contains a considerable amount of data (e.g., CUSTOMER and VENDOR_CONTROL). These tables will need to reviewed with Finance to determine what level of access should be provided to the user community.

Table Name: CUSTOMER

Comments: Snapshot for CFS table CUSTOMER

Columns:

#	Name	Data Type	Nulls?	Default Value
1	CUSTOMER_NO	NUMBER(10)	Y	
2	CUSTOMER_CODE	VARCHAR2(6)	Y	
3	CUSTOMER_NAME	VARCHAR2(40)	Y	
4	ATTENTION_TO	VARCHAR2(40)	Y	
5	ADDRESS1	VARCHAR2(40)	Y	
6	ADDRESS2	VARCHAR2(40)	Y	
7	ADDRESS3	VARCHAR2(40)	Y	
8	CITY	VARCHAR2(25)	Y	
9	ADDRESS_STATE	VARCHAR2(2)	Y	
10	ZIP_CODE	VARCHAR2(10)	Y	
11	COUNTRY_CODE	VARCHAR2(6)	Y	
12	MAIN_PHONE_NO	VARCHAR2(20)	Y	
13	AP_PHONE_NO	VARCHAR2(20)	Y	
14	FAX_NO	VARCHAR2(20)	Y	
15	OTHER_PHONE_NO	VARCHAR2(20)	Y	
16	NOTES	VARCHAR2(240)	Y	
17	CREDIT_LIMIT	NUMBER(14)	Y	
18	BILLING_CYCLE	VARCHAR2(6)	Y	
19	AUTHORIZED_CHARGERS	VARCHAR2(240)	Y	
20	ACTIVE_STATUS	VARCHAR2(1)	Y	
21	STATUS_DATE	DATE	Y	
22	INVOICE_TYPE	VARCHAR2(6)	Y	
23	CUSTOMER_TYPE	VARCHAR2(6)	Y	
24	DELETE_DATE	DATE	Y	
25	ORIGIN_DATE	DATE	Y	
26	NET_DAYS	NUMBER(2)	Y	
27	DISCOUNT_PERCENT	NUMBER(6,3)	Y	
28	DISCOUNT_DAYS	NUMBER(2)	Y	
29	ACCEPT_PARTIALS	VARCHAR2(6)	Y	
30	ACCEPT_BACKORDERS	VARCHAR2(6)	Y	
31	DOMESTIC_FLAG	VARCHAR2(1)	Y	

#	Name	Data Type	Nulls?	Default Value
32	TIN_NO	VARCHAR2(11)	Y	
33	TYPE_1099	VARCHAR2(6)	Y	
34	INTERNAL_FLAG	VARCHAR2(1)	Y	
35	CHARGE_SHIPPING_FLAG	VARCHAR2(1)	Y	
36	FORM_1099_DATE	DATE	Y	
37	FORM_1042_DATE	DATE	Y	
38	INCOME_CODE	VARCHAR2(6)	Y	
39	EXEMPTION_CODE	VARCHAR2(6)	Y	
40	RECIPIENT_TYPE	VARCHAR2(6)	Y	
41	INTERFACE_OPTION_CODE	VARCHAR2(6)	Y	
42	INTERFACE_CUSTOMER_NO	NUMBER(10)	Y	
43	USER_NAME	VARCHAR2(30)	Y	
44	MODIFICATION_DATE	DATE	Y	
45	DEVICE_NAME	VARCHAR2(30)	Y	
46	DOC_BUREAU	VARCHAR2(1)	Y	
47	FEDERAL_AGENCY_CODE	VARCHAR2(6)	Y	

Table Name: CUSTOMER_CONTACT

Comments: Snapshot for CFS table CUSTOMER_CONTACT

Columns:

#	Name	Data Type	Nulls?	Default Value
1	CUSTOMER_NO	NUMBER(10)	Y	
2	CONTACT_NO	NUMBER(6)	Y	
3	CONTACT_TYPE	VARCHAR2(6)	Y	
4	NOTES	VARCHAR2(240)	Y	
5	PHONE_NO	VARCHAR2(20)	Y	
6	FAX_NO	VARCHAR2(20)	Y	
7	DELETE_DATE	DATE	Y	
8	ORIGIN_DATE	DATE	Y	
9	FIRST_NAME	VARCHAR2(20)	Y	
10	MIDDLE_NAME	VARCHAR2(10)	Y	
11	LAST_NAME	VARCHAR2(20)	Y	
12	COMPANY	VARCHAR2(40)	Y	
13	ADDRESS1	VARCHAR2(40)	Y	
14	ADDRESS2	VARCHAR2(40)	Y	
15	ADDRESS3	VARCHAR2(40)	Y	
16	CITY	VARCHAR2(25)	Y	
17	ADDRESS_STATE	VARCHAR2(2)	Y	
18	ZIP_CODE	VARCHAR2(10)	Y	
19	COUNTRY_CODE	VARCHAR2(6)	Y	
20	TITLE	VARCHAR2(40)	Y	
21	AGENCY_LOCATION_CODE	VARCHAR2(10)	Y	
22	APPROP_SYMBOL	VARCHAR2(21)	Y	
23	TAX_BODY	NUMBER(4)	Y	
24	ORG1_CODE	VARCHAR2(2)	Y	
25	ORG2_CODE	VARCHAR2(2)	Y	
26	ORG3_CODE	VARCHAR2(4)	Y	
27	ORG4_CODE	NUMBER(2)	Y	
28	ORG5_CODE	NUMBER(2)	Y	
29	ORG6_CODE	NUMBER(2)	Y	
30	ORG7_CODE	NUMBER(2)	Y	
31	PROJECT_CODE	VARCHAR2(7)	Y	
32	BUREAU_CODE	NUMBER(2)	Y	
33	ACTIVE_STATUS	VARCHAR2(1)	Y	
34	STATUS_DATE	DATE	Y	
35	USER_NAME	VARCHAR2(30)	Y	
36	MODIFICATION_DATE	DATE	Y	
37	DEVICE_NAME	VARCHAR2(30)	Y	
38	TYPE_1099	VARCHAR2(6)	Y	

Table Name: DW_BUREAU_DIM

Comments: This table is used to define the bureau dimension

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_KEY	NUMBER(8)	N	
2	BUREAU_CODE	NUMBER(2)	N	
3	BUREAU_NAME	VARCHAR2(40)	N	

Table Name: DW_EMPLOYEE_DIM

Comments: This table is used to define the employee dimension

#	Name	Data Type	Nulls?	Default Value
1	EMPLOYEE_KEY	NUMBER(8)	N	
2	EMP_NO	NUMBER(6)	N	
3	FIRST_NAME	VARCHAR2(20)	N	
4	LAST_NAME	VARCHAR2(20)	N	
5	MIDDLE_NAME	VARCHAR2(15)	Y	
6	EMAIL_ADDRESS	VARCHAR2(50)	Y	
7	MODIFICATION_DATE	DATE	Y	

Table Name: FUND

Comments: Snapshot for CFS table FUND

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	FUND_CODE	NUMBER(2)	Y	
3	FUND_TITLE	VARCHAR2(40)	Y	
4	FUND_TYPE	VARCHAR2(6)	Y	
5	ACCOUNTING_BASIS	VARCHAR2(6)	Y	
6	AVAILABILITY_CODE	VARCHAR2(6)	Y	
7	BEGIN_DATE	DATE	Y	
8	END_DATE	DATE	Y	
9	PREFIX	NUMBER(2)	Y	
10	SUFFIX	NUMBER(2)	Y	
11	FUND_GROUP	NUMBER(4)	Y	
12	INTERNAL_FLAG	VARCHAR2(1)	Y	
13	APPROP_SYMBOL	VARCHAR2(21)	Y	
14	APPROVED_BY	VARCHAR2(26)	Y	
15	APPROVED_DATE	DATE	Y	
16	APPROVED_EMP_NO	NUMBER(6)	Y	
17	APPROVED_FLAG	VARCHAR2(1)	Y	
18	USER_NAME	VARCHAR2(30)	Y	
19	MODIFICATION_DATE	DATE	Y	
20	DEVICE_NAME	VARCHAR2(30)	Y	
21	ENTITY_FLAG	VARCHAR2(1)	Y	
22	ENTITY_CODE	VARCHAR2(5)	Y	
23	WARNING_PERCENT	NUMBER	Y	

Table Name: OBJECT1

Comments: Snapshot for CFS table OBJECT1

#	Name	Data Type	Nulls?	Default Value
1	OBJECT1_CODE	NUMBER(2)	Y	
2	OBJECT1_DESCR	VARCHAR2(40)	Y	
3	ACTIVE_STATUS	VARCHAR2(1)	Y	
4	STATUS_DATE	DATE	Y	
5	USER_NAME	VARCHAR2(30)	Y	
6	MODIFICATION_DATE	DATE	Y	
7	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: OBJECT2

Comments: Snapshot for CFS table OBJECT2

Columns:

#	Name	Data Type	Nulls?	Default Value
1	OBJECT1_CODE	NUMBER(2)	Y	
2	OBJECT2_CODE	NUMBER(2)	Y	
3	OBJECT2_DESCR	VARCHAR2(40)	Y	
4	ACTIVE_STATUS	VARCHAR2(1)	Y	
5	STATUS_DATE	DATE	Y	
6	USER_NAME	VARCHAR2(30)	Y	
7	MODIFICATION_DATE	DATE	Y	
8	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: OBJECT3

Comments: Snapshot for CFS table OBJECT3

Columns:

#	Name	Data Type	Nulls?	Default Value
1	OBJECT1_CODE	NUMBER(2)	Y	
2	OBJECT2_CODE	NUMBER(2)	Y	
3	OBJECT3_CODE	NUMBER(2)	Y	
4	OBJECT3_DESCR	VARCHAR2(40)	Y	
5	ACTIVE_STATUS	VARCHAR2(1)	Y	
6	STATUS_DATE	DATE	Y	
7	USER_NAME	VARCHAR2(30)	Y	
8	MODIFICATION_DATE	DATE	Y	
9	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: OBJECT4

Comments: Snapshot for CFS table OBJECT4

Columns:

#	Name	Data Type	Nulls?	Default Value
1	OBJECT1_CODE	NUMBER(2)	Y	
2	OBJECT2_CODE	NUMBER(2)	Y	
3	OBJECT3_CODE	NUMBER(2)	Y	
4	OBJECT4_CODE	NUMBER(2)	Y	
5	OBJECT4_DESCR	VARCHAR2(40)	Y	
6	OMB_OBJECT_CLASS	NUMBER(3,1)	Y	
7	CF_CATEGORY	NUMBER(3,1)	Y	
8	GOVERNMENTAL_FLAG	VARCHAR2(1)	Y	
9	ACTIVE_STATUS	VARCHAR2(1)	Y	
10	STATUS_DATE	DATE	Y	
11	USER_NAME	VARCHAR2(30)	Y	
12	MODIFICATION_DATE	DATE	Y	
13	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: ORG1

Comments: Snapshot for CFS table ORG1

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	ORG1_CODE	VARCHAR2(2)	Y	
3	ORG1_DESCR	VARCHAR2(40)	Y	
4	ACTIVE_STATUS	VARCHAR2(1)	Y	
5	STATUS_DATE	DATE	Y	
6	USER_NAME	VARCHAR2(30)	Y	
7	MODIFICATION_DATE	DATE	Y	
8	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: ORG2

Comments: Snapshot for CFS table ORG2

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	ORG1_CODE	VARCHAR2(2)	Y	
3	ORG2_CODE	VARCHAR2(2)	Y	
4	ORG2_DESCR	VARCHAR2(40)	Y	
5	ACTIVE_STATUS	VARCHAR2(1)	Y	
6	STATUS_DATE	DATE	Y	
7	USER_NAME	VARCHAR2(30)	Y	
8	MODIFICATION_DATE	DATE	Y	
9	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: ORG3

Comments: Snapshot for CFS table ORG3

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	ORG1_CODE	VARCHAR2(2)	Y	
3	ORG2_CODE	VARCHAR2(2)	Y	
4	ORG3_CODE	VARCHAR2(4)	Y	
5	ORG3_DESCR	VARCHAR2(40)	Y	
6	ACTIVE_STATUS	VARCHAR2(1)	Y	
7	STATUS_DATE	DATE	Y	
8	USER_NAME	VARCHAR2(30)	Y	
9	MODIFICATION_DATE	DATE	Y	

10	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: ORG4

Comments: Snapshot for CFS table ORG4

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	ORG1_CODE	VARCHAR2(2)	Y	
3	ORG2_CODE	VARCHAR2(2)	Y	
4	ORG3_CODE	VARCHAR2(4)	Y	
5	ORG4_CODE	NUMBER(2)	Y	
6	ORG4_DESCR	VARCHAR2(40)	Y	
7	ACTIVE_STATUS	VARCHAR2(1)	Y	
8	STATUS_DATE	DATE	Y	
9	USER_NAME	VARCHAR2(30)	Y	
10	MODIFICATION_DATE	DATE	Y	
11	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: ORG5

Comments: Snapshot for CFS table ORG5

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	ORG1_CODE	VARCHAR2(2)	Y	
3	ORG2_CODE	VARCHAR2(2)	Y	
4	ORG3_CODE	VARCHAR2(4)	Y	
5	ORG4_CODE	NUMBER(2)	Y	
6	ORG5_CODE	NUMBER(2)	Y	
7	ORG5_DESCR	VARCHAR2(40)	Y	
8	ACTIVE_STATUS	VARCHAR2(1)	Y	
9	STATUS_DATE	DATE	Y	
10	USER_NAME	VARCHAR2(30)	Y	
11	MODIFICATION_DATE	DATE	Y	
12	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: ORG6

Comments: Snapshot for CFS table ORG6

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	ORG1_CODE	VARCHAR2(2)	Y	
3	ORG2_CODE	VARCHAR2(2)	Y	
4	ORG3_CODE	VARCHAR2(4)	Y	
5	ORG4_CODE	NUMBER(2)	Y	
6	ORG5_CODE	NUMBER(2)	Y	
7	ORG6_CODE	NUMBER(2)	Y	
8	ORG6_DESCR	VARCHAR2(40)	Y	
9	ACTIVE_STATUS	VARCHAR2(1)	Y	
10	STATUS_DATE	DATE	Y	
11	USER_NAME	VARCHAR2(30)	Y	
12	MODIFICATION_DATE	DATE	Y	
13	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: ORG7

Comments: Snapshot for CFS table ORG7

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	ORG1_CODE	VARCHAR2(2)	Y	
3	ORG2_CODE	VARCHAR2(2)	Y	
4	ORG3_CODE	VARCHAR2(4)	Y	
5	ORG4_CODE	NUMBER(2)	Y	
6	ORG5_CODE	NUMBER(2)	Y	
7	ORG6_CODE	NUMBER(2)	Y	
8	ORG7_CODE	NUMBER(2)	Y	
9	ORG7_DESCR	VARCHAR2(40)	Y	
10	ACTIVE_STATUS	VARCHAR2(1)	Y	
11	STATUS_DATE	DATE	Y	
12	USER_NAME	VARCHAR2(30)	Y	
13	MODIFICATION_DATE	DATE	Y	
14	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: DW_PAYMENT_OFFICE_DIM

Comments: This table is used to define the payment office dimension

Columns:

#	Name	Data Type	Nulls?	Default Value
1	PAYMENT_OFFICE_KEY	NUMBER(8)	N	
2	PAYMENT_OFFICE_CODE	VARCHAR2(6)	N	
3	PAYMENT_OFFICE_NAME	VARCHAR2(40)	N	

Table Name: PROGRAM1

Comments: snapshot table for snapshot CAMSADM.PROGRAM1

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	FUND_CODE	NUMBER(2)	Y	
3	PROGRAM1_CODE	NUMBER(2)	Y	
4	PROGRAM1_DESCR	VARCHAR2(40)	Y	
5	ACTIVE_STATUS	VARCHAR2(1)	Y	
6	STATUS_DATE	DATE	Y	
7	USER_NAME	VARCHAR2(30)	Y	
8	MODIFICATION_DATE	DATE	Y	
9	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: PROGRAM2

Comments: Snapshot for CFS table PROGRAM2

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	FUND_CODE	NUMBER(2)	Y	
3	PROGRAM1_CODE	NUMBER(2)	Y	
4	PROGRAM2_CODE	NUMBER(2)	Y	
5	PROGRAM2_DESCR	VARCHAR2(40)	Y	
6	ACTIVE_STATUS	VARCHAR2(1)	Y	
7	STATUS_DATE	DATE	Y	
8	USER_NAME	VARCHAR2(30)	Y	
9	MODIFICATION_DATE	DATE	Y	
10	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: PROGRAM3

Comments: Snapshot for CFS table PROGRAM3

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	FUND_CODE	NUMBER(2)	Y	
3	PROGRAM1_CODE	NUMBER(2)	Y	
4	PROGRAM2_CODE	NUMBER(2)	Y	
5	PROGRAM3_CODE	NUMBER(2)	Y	
6	PROGRAM3_DESCR	VARCHAR2(40)	Y	
7	ACTIVE_STATUS	VARCHAR2(1)	Y	
8	STATUS_DATE	DATE	Y	
9	USER_NAME	VARCHAR2(30)	Y	
10	MODIFICATION_DATE	DATE	Y	
11	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: PROGRAM4

Comments: Snapshot for CFS table PROGRAM4

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	FUND_CODE	NUMBER(2)	Y	
3	PROGRAM1_CODE	NUMBER(2)	Y	
4	PROGRAM2_CODE	NUMBER(2)	Y	
5	PROGRAM3_CODE	NUMBER(2)	Y	
6	PROGRAM4_CODE	NUMBER(3)	Y	
7	PROGRAM4_DESCR	VARCHAR2(40)	Y	
8	ACTIVE_STATUS	VARCHAR2(1)	Y	
9	STATUS_DATE	DATE	Y	
10	USER_NAME	VARCHAR2(30)	Y	
11	MODIFICATION_DATE	DATE	Y	
12	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: PROJECT

Comments: Snapshot for CFS table PROJECT

Columns:

#	Name	Data Type	Nulls?	Default Value
1	FUND_CODE	NUMBER(2)	Y	
2	BUREAU_CODE	NUMBER(2)	Y	
3	PROGRAM1_CODE	NUMBER(2)	Y	
4	PROGRAM2_CODE	NUMBER(2)	Y	
5	PROGRAM3_CODE	NUMBER(2)	Y	
6	PROGRAM4_CODE	NUMBER(3)	Y	
7	PROJECT_CODE	VARCHAR2(7)	Y	
8	PROJECT_DESCR	VARCHAR2(40)	Y	
9	PROJECT_TYPE	VARCHAR2(6)	Y	
10	MANAGER_EMP_NO	NUMBER(6)	Y	
11	ADMIN_EMP_NO	NUMBER(6)	Y	
12	TECHREP_EMP_NO	NUMBER(6)	Y	
13	POC_EMP_NO	NUMBER(6)	Y	
14	SCIENCE_CODE	VARCHAR2(6)	Y	
15	NSF_CODE	VARCHAR2(6)	Y	
16	WORK_SITE	VARCHAR2(6)	Y	
17	PRIOR_PROJECT_CODE	VARCHAR2(7)	Y	
18	DIRECT_FLAG	VARCHAR2(1)	Y	
19	BEGIN_DATE	DATE	Y	
20	END_DATE	DATE	Y	
21	NOTES	VARCHAR2(240)	Y	
22	S_ORG1_CODE	VARCHAR2(2)	Y	
23	S_ORG2_CODE	VARCHAR2(2)	Y	
24	S_ORG3_CODE	VARCHAR2(4)	Y	
25	S_ORG4_CODE	NUMBER(2)	Y	
26	S_ORG5_CODE	NUMBER(2)	Y	
27	S_ORG6_CODE	NUMBER(2)	Y	
28	S_ORG7_CODE	NUMBER(2)	Y	
29	P_ORG1_CODE	VARCHAR2(2)	Y	
30	P_ORG2_CODE	VARCHAR2(2)	Y	
31	P_ORG3_CODE	VARCHAR2(4)	Y	
32	P_ORG4_CODE	NUMBER(2)	Y	
33	P_ORG5_CODE	NUMBER(2)	Y	
34	P_ORG6_CODE	NUMBER(2)	Y	
35	P_ORG7_CODE	NUMBER(2)	Y	
36	BUDGET_INITIATIVE	VARCHAR2(5)	Y	
37	APPROVED_FLAG	VARCHAR2(1)	Y	
38	APPROVED_EMP_NO	NUMBER(6)	Y	
39	APPROVED_BY	VARCHAR2(30)	Y	
40	APPROVED_DATE	DATE	Y	
41	USER_DEFINE_ACCS	NUMBER(6)	Y	

#	Name	Data Type	Nulls?	Default Value
42	ACTIVE_STATUS	VARCHAR2(1)	Y	
43	STATUS_DATE	DATE	Y	
44	STAT_UNIT_QTY	NUMBER(8,2)	Y	
45	LABOR_ESTIMATE	NUMBER(13,2)	Y	
46	OTHER_ESTIMATE	NUMBER(13,2)	Y	
47	TOTAL_AUTHORIZATION	NUMBER(13,2)	Y	
48	PROJECT_LEADER	VARCHAR2(22)	Y	
49	BASE_FLAG	VARCHAR2(1)	Y	
50	INTERFACE_OPTION_CODE	VARCHAR2(6)	Y	
51	USER_NAME	VARCHAR2(30)	Y	
52	MODIFICATION_DATE	DATE	Y	
53	DEVICE_NAME	VARCHAR2(30)	Y	
54	PRODUCTION_FLAG	VARCHAR2(1)	Y	
55	THEME_CODE	VARCHAR2(2)	Y	
56	GOAL_CODE	VARCHAR2(2)	Y	

Table Name: TASK

Comments: Snapshot for CFS table TASK

Columns:

#	Name	Data Type	Nulls?	Default Value
1	BUREAU_CODE	NUMBER(2)	Y	
2	PROJECT_CODE	VARCHAR2(7)	Y	
3	TASK_CODE	VARCHAR2(3)	Y	
4	TASK_DESCR	VARCHAR2(40)	Y	
5	BEGIN_DATE	DATE	Y	
6	END_DATE	DATE	Y	
7	NOTES	VARCHAR2(240)	Y	
8	BE_STATUS	VARCHAR2(1)	Y	
9	PR_STATUS	VARCHAR2(1)	Y	
10	PO_STATUS	VARCHAR2(1)	Y	
11	GJ_STATUS	VARCHAR2(1)	Y	
12	FA_STATUS	VARCHAR2(1)	Y	
13	AP_STATUS	VARCHAR2(1)	Y	
14	AR_STATUS	VARCHAR2(1)	Y	
15	LB_STATUS	VARCHAR2(1)	Y	
16	TA_STATUS	VARCHAR2(1)	Y	
17	ACTIVE_STATUS	VARCHAR2(1)	Y	
18	STATUS_DATE	DATE	Y	
19	USER_NAME	VARCHAR2(30)	Y	
20	MODIFICATION_DATE	DATE	Y	
21	DEVICE_NAME	VARCHAR2(30)	Y	

Table Name: VENDOR_CONTROL

Comments: Snapshot for CFS table VENDOR_CONTROL

Columns:

#	Name	Data Type	Nulls?	Default Value
1	VENDOR_NO	NUMBER(10)	Y	
2	VENDOR_CODE	VARCHAR2(9)	Y	
3	VENDOR_TYPE	VARCHAR2(6)	Y	
4	ORIGIN_DATE	DATE	Y	
5	STATUS_DATE	DATE	Y	
6	ACTIVE_STATUS	VARCHAR2(1)	Y	
7	DELETE_DATE	DATE	Y	
8	SIC_CODE	VARCHAR2(10)	Y	
9	LABOR_SURPLUS_FLAG	VARCHAR2(1)	Y	
10	MINORITY_OWNED_FLAG	VARCHAR2(1)	Y	
11	MINORITY_CODE	VARCHAR2(6)	Y	
12	WOMAN_OWNED_FLAG	VARCHAR2(1)	Y	
13	REPORT_NAME	VARCHAR2(30)	Y	
14	SIZE_FLAG	VARCHAR2(2)	Y	
15	VENDOR_ACCOUNT_NO	VARCHAR2(20)	Y	
16	NET_DAYS1	NUMBER(2)	Y	
17	DISCOUNT_FLAG1	VARCHAR2(1)	Y	
18	DISCOUNT_AMOUNT1	NUMBER(8,3)	Y	
19	DISCOUNT_DAYS1	NUMBER(2)	Y	
20	NET_DAYS2	NUMBER(2)	Y	
21	DISCOUNT_FLAG2	VARCHAR2(1)	Y	
22	DISCOUNT_AMOUNT2	NUMBER(8,3)	Y	
23	DISCOUNT_DAYS2	NUMBER(2)	Y	
24	FOB_POINT	VARCHAR2(8)	Y	
25	NOTES	VARCHAR2(240)	Y	
26	ASSIGNMENT_FLAG	VARCHAR2(1)	Y	
27	ASSIGNMENT_DATE	DATE	Y	
28	FEDERAL_AGENCY_CODE	VARCHAR2(6)	Y	
29	INTERFACE_OPTION_CODE	VARCHAR2(6)	Y	
30	FOREIGN_FLAG	VARCHAR2(1)	Y	
31	VENDOR_DIVISION	VARCHAR2(25)	Y	
32	COMMON_PARENT_TIN	VARCHAR2(9)	Y	
33	COMMON_PARENT_NAME	VARCHAR2(30)	Y	
34	USER_NAME	VARCHAR2(30)	Y	
35	MODIFICATION_DATE	DATE	Y	
36	DEVICE_NAME	VARCHAR2(30)	Y	
37	DOC_BUREAU	VARCHAR2(1)	Y	
38	INTERNAL_FLAG	VARCHAR2(1)	Y	

Table Name: VENDOR_DETAIL

Comments: snapshot table for snapshot CAMSADM.VENDOR_DETAIL

Columns:

#	Name	Data Type	Nulls?	Default Value
1	VENDOR_NO	NUMBER(10)	N	
2	VENDOR_ID	NUMBER(6)	N	
3	ADDRESS_TYPE	VARCHAR2(6)	N	
4	ADDRESS_NAME	VARCHAR2(30)	Y	
5	ADDRESS1	VARCHAR2(30)	Y	
6	ADDRESS2	VARCHAR2(30)	Y	
7	CITY	VARCHAR2(20)	Y	
8	ADDRESS_STATE	VARCHAR2(2)	Y	
9	ZIP_CODE	VARCHAR2(10)	Y	
10	COUNTRY_CODE	VARCHAR2(6)	Y	
11	ACTIVE_STATUS	VARCHAR2(1)	Y	
12	STATUS_DATE	DATE	Y	
13	DELETE_DATE	DATE	Y	
14	CONTACT	VARCHAR2(30)	Y	
15	PHONE_NO	VARCHAR2(20)	Y	
16	FAX_NO	VARCHAR2(20)	Y	
17	TYPE_1099	VARCHAR2(6)	Y	
18	FORM_1099_DATE	DATE	Y	
19	FORM_1042_FLAG	VARCHAR2(1)	Y	
20	FORM_1042_DATE	DATE	Y	
21	INCOME_CODE	VARCHAR2(6)	Y	
22	EXEMPTION_CODE	VARCHAR2(6)	Y	
23	RECIPIENT_TYPE	VARCHAR2(6)	Y	
24	FEIN_NO	VARCHAR2(11)	Y	
25	DUNS_NO	VARCHAR2(9)	Y	
26	WITHHOLD_FLAG	VARCHAR2(1)	Y	
27	ENTITY_TYPE	VARCHAR2(6)	Y	
28	W9_RECEIVED	VARCHAR2(1)	Y	
29	PROMPT_PAY_FLAG	VARCHAR2(1)	Y	
30	PAYMENT_METHOD	VARCHAR2(6)	Y	
31	PAYMENT_COUNTRY_CODE	VARCHAR2(6)	Y	
32	EFT_NO	NUMBER(9)	Y	
33	EFT_ACCOUNT_NO	VARCHAR2(20)	Y	
34	EFT_ACCOUNT_TYPE	VARCHAR2(6)	Y	
35	PRE_NOTE_FLAG	VARCHAR2(1)	Y	
36	NOTES	VARCHAR2(240)	Y	
37	USER_NAME	VARCHAR2(30)	Y	
38	MODIFICATION_DATE	DATE	Y	
39	DEVICE_NAME	VARCHAR2(30)	Y	
40	APP_CODE	VARCHAR2(8)	Y	

Appendix E – CAMS to FIMA Code Translation

This appendix provides the mapping of various FIMA code values to their CAMS equivalent.

This table relates the FIMA Country Codes to the CFS Country Codes.

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
CNTRY	AD	140	Y
CNTRY	AE	888	Y
CNTRY	AF	110	Y
CNTRY	AG	149	Y
CNTRY	AI	142	Y
CNTRY	AL	120	Y
CNTRY	AM	151	Y
CNTRY	AN	640	Y
CNTRY	AO	141	Y
CNTRY	AQ	143	Y
CNTRY	AR	150	Y
CNTRY	AS	60	Y
CNTRY	AT	165	Y
CNTRY	AU	160	Y
CNTRY	AW	152	Y
CNTRY	AZ	167	Y
CNTRY	BB	184	Y
CNTRY	BD	182	Y
CNTRY	BE	190	Y
CNTRY	BF	927	Y
CNTRY	BG	245	Y
CNTRY	BH	181	Y
CNTRY	BI	252	Y
CNTRY	BJ	311	Y
CNTRY	BM	195	Y
CNTRY	BN	232	Y
CNTRY	BO	205	Y
CNTRY	BP	797	Y
CNTRY	BR	220	Y
CNTRY	BS	180	Y
CNTRY	BT	200	Y
CNTRY	BV	212	Y
CNTRY	BW	210	Y
CNTRY	BY	253	Y
CNTRY	BZ	227	Y
CNTRY	CA	260	Y
CNTRY	CC	284	Y
CNTRY	CF	269	Y
CNTRY	CG	290	Y

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
CNTRY	CH	855	Y
CNTRY	CI	485	Y
CNTRY	CK	292	Y
CNTRY	CL	275	Y
CNTRY	CM	257	Y
CNTRY	CN	280	Y
CNTRY	CO	285	Y
CNTRY	CR	295	Y
CNTRY	CS	310	Y
CNTRY	CU	300	Y
CNTRY	CV	264	Y
CNTRY	CX	282	Y
CNTRY	CY	305	Y
CNTRY	CZ	310	Y
CNTRY	DE	394	Y
CNTRY	DJ	317	Y
CNTRY	DK	315	Y
CNTRY	DM	318	Y
CNTRY	DO	320	Y
CNTRY	DZ	125	Y
CNTRY	EC	325	Y
CNTRY	EE	333	Y
CNTRY	EG	327	Y
CNTRY	EH	947	Y
CNTRY	ES	830	Y
CNTRY	ET	335	Y
CNTRY	FI	340	Y
CNTRY	FJ	338	Y
CNTRY	FK	337	Y
CNTRY	FM	339	Y
CNTRY	FO	336	Y
CNTRY	FR	350	Y
CNTRY	FX	368	Y
CNTRY	GA	388	Y
CNTRY	GB	925	Y
CNTRY	GD	406	Y
CNTRY	GE	395	Y
CNTRY	GF	418	Y
CNTRY	GH	396	Y
CNTRY	GI	397	Y
CNTRY	GL	405	Y
CNTRY	GM	389	Y
CNTRY	GN	417	Y
CNTRY	GP	407	Y
CNTRY	GQ	332	Y
CNTRY	GR	400	Y

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
CNTRY	GS	826	Y
CNTRY	GT	415	Y
CNTRY	GU	66	Y
CNTRY	GW	737	Y
CNTRY	GY	418	Y
CNTRY	HK	435	Y
CNTRY	HM	421	Y
CNTRY	HN	430	Y
CNTRY	HT	420	Y
CNTRY	HU	445	Y
CNTRY	ID	458	Y
CNTRY	IE	470	Y
CNTRY	IL	475	Y
CNTRY	IN	455	Y
CNTRY	IO	222	Y
CNTRY	IQ	465	Y
CNTRY	IR	460	N
CNTRY	IS	450	Y
CNTRY	IT	480	Y
CNTRY	JM	487	Y
CNTRY	JO	500	Y
CNTRY	JP	490	Y
CNTRY	KE	505	Y
CNTRY	KG	521	Y
CNTRY	KH	255	Y
CNTRY	KI	62	Y
CNTRY	KM	287	Y
CNTRY	KN	759	Y
CNTRY	KP	515	Y
CNTRY	KR	516	Y
CNTRY	KW	520	Y
CNTRY	KY	268	Y
CNTRY	KZ	503	Y
CNTRY	LA	530	Y
CNTRY	LB	540	Y
CNTRY	LC	770	Y
CNTRY	LI	553	Y
CNTRY	LK	272	Y
CNTRY	LR	545	Y
CNTRY	LS	543	Y
CNTRY	LT	555	Y
CNTRY	LU	570	Y
CNTRY	LV	531	Y
CNTRY	LY	550	Y
CNTRY	MA	610	Y
CNTRY	MC	607	Y

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
CNTRY	MD	598	Y
CNTRY	MG	575	Y
CNTRY	MH	596	Y
CNTRY	ML	585	Y
CNTRY	MN	608	Y
CNTRY	MO	571	Y
CNTRY	MP	673	Y
CNTRY	MQ	591	Y
CNTRY	MR	592	Y
CNTRY	MS	609	Y
CNTRY	MT	590	Y
CNTRY	MU	593	Y
CNTRY	MV	581	Y
CNTRY	MW	577	Y
CNTRY	MX	595	Y
CNTRY	MY	580	Y
CNTRY	MZ	615	Y
CNTRY	NA	821	Y
CNTRY	NC	645	Y
CNTRY	NE	667	Y
CNTRY	NF	672	Y
CNTRY	NG	670	Y
CNTRY	NI	665	Y
CNTRY	NL	630	Y
CNTRY	NO	685	Y
CNTRY	NP	625	Y
CNTRY	NR	618	Y
CNTRY	NU	671	Y
CNTRY	NZ	660	Y
CNTRY	OM	616	Y
CNTRY	PA	710	Y
CNTRY	PE	720	Y
CNTRY	PF	367	Y
CNTRY	PG	712	Y
CNTRY	PH	725	Y
CNTRY	PK	700	Y
CNTRY	PL	730	Y
CNTRY	PM	771	Y
CNTRY	PN	726	Y
CNTRY	PR	72	Y
CNTRY	PT	735	Y
CNTRY	PW	75	Y
CNTRY	PY	715	Y
CNTRY	QA	747	Y
CNTRY	RE	750	Y
CNTRY	RO	755	Y

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
CNTRY	RU	756	Y
CNTRY	RW	758	Y
CNTRY	SA	785	Y
CNTRY	SB	797	Y
CNTRY	SC	788	Y
CNTRY	SD	835	Y
CNTRY	SE	850	Y
CNTRY	SG	795	Y
CNTRY	SH	765	Y
CNTRY	SL	790	Y
CNTRY	SM	782	Y
CNTRY	SN	787	Y
CNTRY	SO	800	Y
CNTRY	SR	840	Y
CNTRY	ST	783	Y
CNTRY	SU	825	Y
CNTRY	SV	330	Y
CNTRY	SY	858	Y
CNTRY	SZ	847	Y
CNTRY	TC	907	Y
CNTRY	TD	273	Y
CNTRY	TF	368	Y
CNTRY	TG	883	Y
CNTRY	TH	875	Y
CNTRY	TJ	860	Y
CNTRY	TK	885	Y
CNTRY	TM	906	Y
CNTRY	TN	890	Y
CNTRY	TO	886	Y
CNTRY	TR	905	Y
CNTRY	TT	887	Y
CNTRY	TV	908	Y
CNTRY	TW	859	Y
CNTRY	TZ	865	Y
CNTRY	UA	911	Y
CNTRY	UG	910	Y
CNTRY	UK	925	Y
CNTRY	UM	75	Y
CNTRY	US	926	Y
CNTRY	UY	930	Y
CNTRY	UZ	931	Y
CNTRY	VA	933	Y
CNTRY	VC	775	Y
CNTRY	VE	940	Y
CNTRY	VG	231	Y
CNTRY	VI	78	Y

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
CNTRY	VN	944	Y
CNTRY	VU	651	Y
CNTRY	WF	943	Y
CNTRY	WS	963	Y
CNTRY	YE	965	Y
CNTRY	YT	597	Y
CNTRY	YU	970	Y
CNTRY	ZA	801	Y
CNTRY	ZM	990	Y
CNTRY	ZR	291	Y
CNTRY	ZW	818	Y

This table relates the FIMA FAC codes to the CFS equivalent.

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
FAC	1240	AGG	N
FAC	ABMC	ABM	Y
FAC	AC	AAA	Y
FAC	ACDA	XXX	N
FAC	ADC	ADC	Y
FAC	ADF	XXX	N
FAC	AEC	EEE	N
FAC	AF	DAF	Y
FAC	AID	AID	Y
FAC	ARC	ARC	Y
FAC	ARMY	DDA	Y
FAC	ATF	TREAS	N
FAC	BIA	XXX	N
FAC	BLM	XXX	N
FAC	BOM	XXX	N
FAC	BOR	XXX	N
FAC	BXA	BXA	Y
FAC	CCR	XXX	N
FAC	CEN	CCB	Y
FAC	CFTC	XXX	N
FAC	CG	TTT	Y
FAC	CIA	CIA	Y
FAC	CNS	CNS	Y
FAC	COE	DDC	Y
FAC	COURTS	AOC	Y
FAC	CPSC	CPS	Y
FAC	DOC	SEC	Y
FAC	DOD	DDD	Y
FAC	DOE	EEN	Y
FAC	DOI	KKK	Y
FAC	DOJ	GGG	Y
FAC	DOL	LLL	Y
FAC	DOT	TTT	Y
FAC	ECRC	XXX	N
FAC	EDA	EDA	Y
FAC	EDUC	EDU	Y
FAC	EEOC	EEC	Y
FAC	EIB	EIB	Y
FAC	EPA	EPA	Y
FAC	ERDA	DDE	N
FAC	ESA	ESA	Y
FAC	EXP	EXP	Y
FAC	FAA	TTA	N
FAC	FBI	XXX	N
FAC	FBOP	XXX	N

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
FAC	FCA	FCA	Y
FAC	FCC	FCC	Y
FAC	FDA	XXX	N
FAC	FDIC	FDI	Y
FAC	FEC	XXX	N
FAC	FEMA	FEM	Y
FAC	FERC	XXX	N
FAC	FHFB	XXX	N
FAC	FHWA	XXX	N
FAC	FLRA	FLR	Y
FAC	FMC	CCM	Y
FAC	FMCS	FMX	Y
FAC	FRSB	XXX	N
FAC	FRTIB	XXX	N
FAC	FS	AGF	N
FAC	FTA	XXX	N
FAC	FTC	FTC	Y
FAC	FWA	KKA	N
FAC	GAO	GAO	Y
FAC	GPO	GPP	Y
FAC	GSA	GPA	Y
FAC	HHS	WWW	Y
FAC	HUD	BBB	Y
FAC	IAF	XXX	N
FAC	IBWC	IBW	Y
FAC	ICC	XXX	N
FAC	INS	XXX	N
FAC	IRS	XXX	N
FAC	ITA	ITA	Y
FAC	ITC	ITC	Y
FAC	LOC	MMM	Y
FAC	MBDA	MBA	Y
FAC	MMS	XXX	N
FAC	MSHA	XXX	N
FAC	MSPB	XXX	N
FAC	NARA	XXX	N
FAC	NASA	NAA	Y
FAC	NAVY	DDN	Y
FAC	NCAR	XXX	N
FAC	NCPC	XXX	N
FAC	NCUA	XXX	N
FAC	NEA	XXX	N
FAC	NEH	XXX	N
FAC	NFAH	XXX	N
FAC	NIH	NIH	N
FAC	NIST	CCA	Y

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
FAC	NLRB	NLB	Y
FAC	NMB	XXX	N
FAC	NOAA	NNA	Y
FAC	NONE	XXX	N
FAC	NPCA	XXX	N
FAC	NPR	XXX	N
FAC	NPS	KKB	N
FAC	NRC	NRC	Y
FAC	NSA	XXX	N
FAC	NSF	EEF	Y
FAC	NTIA	NTI	Y
FAC	NTIS	TIS	Y
FAC	NTO	NTO	N
FAC	NTSB	XXX	N
FAC	OGE	XXX	N
FAC	OIG	OIG	Y
FAC	OPIC	OPI	Y
FAC	OPM	CSC	Y
FAC	OSC	XXX	N
FAC	OSHA	XXX	N
FAC	OSM	XXX	N
FAC	OSTP	XXX	N
FAC	PBGC	XXX	N
FAC	PCC	PCC	Y
FAC	PTO	PTO	Y
FAC	RRB	XXX	N
FAC	RTC	XXX	N
FAC	SACA	XXX	N
FAC	SBA	SBA	Y
FAC	SEC	XXX	N
FAC	SI	SIX	Y
FAC	SSA	SSA	Y
FAC	SSS	SSX	Y
FAC	STATE	NNN	Y
FAC	TA	CTA	Y
FAC	TDA	XXX	N
FAC	TDPOB	XXX	N
FAC	TREAS	RRR	Y
FAC	TVA	EEG	Y
FAC	USDA	AGG	Y
FAC	USGS	XXX	N
FAC	USHOR	HOP	Y
FAC	USIA	EEK	Y
FAC	USPS	PPP	Y
FAC	USTAX	TCX	Y
FAC	USTTA	XXX	N

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
FAC	VA	VAX	Y
FAC	VOA	XXX	N
FAC	WRC	XXX	N

This relates the FIMA Late Code values to the CFS equivalent.

CODE_TYPE	CODE_VALUE	FIMA_CODE	ACTIVE_STATUS
LATE	1A	00B	Y
LATE	1B	00C	Y
LATE	1C	00A	Y
LATE	2A	00E	Y
LATE	2B	00F	Y
LATE	2C	00G	Y
LATE	2D	OOI	Y

This table relates the FIMA DOCUMENT_TYPE codes to the CFS Document Codes.

TRANS_NO	FIMA_DOCUMENT_TYPE	FIMA_DOC_CLASS	FIMA_DOCUMENT_DESCR	SUB SYSTEM_ID	CFS_DOC1_CODE	CFS_DOC2_CODE	ACTIVE_STATUS	FIMA_2_REF_FLAG	FIMA_2_CONTROL_FLAG
3	1	A	INTERAGENCY AGREEMENT PURCHASE ORDER	U	IAGPO		Y	S	F
144	1	B	INTERFACE ABORT CORRECTION	P	NOMTCH	INTAGR	N	S	F
3713	1	B	INTERAGENCY AGREEMENT PURCHASE AGREEMENT	P	VINV	INTAGR	Y	S	F
2852	1	B	INTERAGENCY AGREEMENT PURCHASE ORDER	P	OPAC	INTAGR	Y	S	F
358	2	A	BLANKET PURCHASE AGREEMENT CALL	U	POCALL		Y	R	Y
415	2	A	FINANCE OFFICE BLANKET PURCHASE AGREEMET	U	FOBPA		Y	S	F
4	2	B	EXPRESS SMALL PACKAGING SERVICES	P	NOMTCH	FEDEX	Y	S	Y
4013	2	B	CFS TO FIMA ERROR CORRECTION	P	RECUR	FEDEX	N	S	Y
4336	2	B	CFS TO FIMA ERROR CORRECTION	P	RECUR	SUPPLY	N	S	F
4014	2	B	CFS TO FIMA ERROR CORRECTION	P	VINV	EXMAIL	N	S	Y
2954	2	B	INTERFACE ABORT CORRECTION	P	NOMTCH	GOODS	N	P	Y
2493	2	B	BLANKET PURCHASE AGREEMENT (OPAC)	P	OPAC	SUPPLY	Y	S	F
386	2	B	EXPRESS MAIL TRANSPORTATION	P	NOMTCH	EXMAIL	Y	S	Y
145	2	B	BLANKET PURCHASE AGREEMENT	P	NOMTCH	SUPPLY	Y	S	F
27	2	B	BLANKET PURCHASE AGREEMENT	P	NOMTCH	SERV	Y	S	F
467	2	B	BLANKET PURCHASE AGREEMENT (OPAC)	P	OPAC	SERV	Y	S	F
6	3	A	NON-RECURRING CONTRACTS	U	CONTRT		Y	S	F
7	3	A	INTERGOVERNMENTAL PERSONNEL ACT ASSIGNM	U	IPA		Y	S	Y
2	5	A	RECURRING DELIVERIES	U	RECDEL		Y	S	F
8	6	B	LEASES (RECUR)	P	RECUR	LEASE	Y	S	F
2209	6	B	LEASES (NOMTCH)	P	NOMTCH	LEASE	Y	S	F

TRANS_NO	FIMA_DOCUMENT_TYPE	FIMA_DOC_CLASS	FIMA_DOCUMENT_DESCR	SUB SYSTEM_ID	CFS_DOC1_CODE	CFS_DOC2_CODE	ACTIVE_STATUS	FIMA_2_REF_FLAG	FIMA_2_CONTROL_FLAG
4741	6	B	CFS TO FIMA ERROR CORRECTION	P	RECUR	ADV	Y	S	F
281	6	B	UTILITIES (RECUR)	P	RECUR	UTIL	Y	S	F
24	6	B	UTILITIES (NOMTCH)	P	NOMTCH	UTIL	Y	S	F
1	7	A	GSA REIMBURSABLE WORK AUTHORIZATION	U	GSARWA		Y	S	F
10	8	A	TRAINING	U	TRAIN		Y	S	F
4214	8	B	TRAINING - ERROR 9/19	P	VINV	TRAIN	N	S	F
4885	8	B	TRAIN: CORRECT CFS2FIMA	P	NOMTCH	TRAIN	Y	S	F
12	12	B	PURCHASE ORDER/INVOICE/VOUCHER	P	NOMTCH	SF44	Y	S	F
2853	13	A	MILSTRIP	U	MILSTR		Y	S	F
4696	13	A	FEDSTRIP	U	FEDSTR		Y	S	F
4697	13	B	FEDSTRIP (NOMTCH)	P	OPAC	FEDSTR	Y	S	G
4698	13	B	MILSTRIP (NOMTCH)	P	OPAC	MILSTR	Y	S	G
4699	13	B	MILSTRIP (NOMTCH)	P	VINV	MILSTR	Y	S	G
1589	14	D	PURCHASE CARD	P	NOMTCH	PCARD	Y	S	G
14	19	B	MISCELLANEOUS CERTIFIED INVOICE	P	NOMTCH	MISC	Y	I	Y
16	19	B	ADVERTISING ORDER	P	NOMTCH	SF1143	Y	I	Y
4496	19	B	MISCELLANEOUS OPAC INVOICE	P	OPAC	MISC	Y	S	F
15	19	B	ARBITRATION FEES	P	NOMTCH	ARBFEE	Y	I	Y
30	20	B	TRAVEL DOMESTIC	P	TRAVNM	DOM	Y	I	F
2893	20	B	INTERFACE ABORT CORRECTION	P	NOMTCH	DOM	Y	I	F
262	20	C	GTA DOMESTIC	P	NOMTCH	GTADOM	Y	S	G
23	23	A	GOVERNMENT BILL OF LADING	U	TRANSP		Y	S	F
263	26	C	GTA PCS	P	NOMTCH	GTAPCS	Y	S	G
31	27	B	TRAVEL FOREIGN	P	TRAVNM	FOR	Y	I	F
264	27	C	GTA FOREIGN	P	NOMTCH	GTAFOR	Y	S	G
388	27	C	INTERFACE ABORT CORRECTION	P	TRAVNM	GTAFOR	N	S	G
4656	29	C	GSA MOTOR POOL	P	NOMTCH	MTRPL	Y	S	G

TRANS_NO	FIMA_DOCUMENT_TYPE	FIMA_DOC_CLASS	FIMA_DOCUMENT_DESCR	SUB SYSTEM_ID	CFS_DOC1_CODE	CFS_DOC2_CODE	ACTIVE_STATUS	FIMA_2_REF_FLAG	FIMA_2_CONTROL_FLAG
21	30	B	RECOMMENDATION FOR RECOGNITION	P	NOMTCH	CD326	Y	V	F
303	30	B	INTERFACE ABORT CORRECTION	P	CM	CD326	N	V	F
67	30	B	IMPREST FUND PURCHASES	P	NOMTCH	NF34-6	Y	P	Y
20	31	B	CLAIM FOR REIMBURSEMENT FOR EXPENDITURE	P	NOMTCH	SF1164	Y	I	Y
2653	31	B	INTERFACE ABORT CORRECTION	P	TRAVNM	SF1164	Y	I	Y
244	33	B	CONTRACT OBSERVER	P	NOMTCH	OBSERV	Y	S	F
1209	33	B	INTERFACE ABORT CORRECTION	P	NFCAP	OBSERV	N	S	F
1449	33	B	INTERFACE ABORT CORRECTION	P	TRANSP	OBSERV	N	S	F
9	35	A	FOREIGN TRAINEE VOUCHER	U	FORTRN		Y	S	Y
242	35	B	INTERFACE ABORT CORRECTION	P	NOMTCH	STIPEN	N	P	Y
28	36	B	OIL COMPANY INVOICES FOR CREDIT CARD	P	NOMTCH	GASCD	Y	S	Y
354	40	A	ONE-TIME PAYMENT PURCHASE ORDER	U	PO		Y	P	Y
204	40	B	INTERFACE ABORT CORRECTION	P	VINV	GOODS	N	P	Y
4743	40	B	INTERFACE ABORT CORRECTION	P	VINV	SERV	Y	P	Y
355	41	A	ADVANCE PAYMENT PURCHASE ORDER	U	POAVPY		Y	P	Y
357	42	A	RECURRING PAYMENT PURCHASE ORDER	U	PORCUR		Y	P	Y
245	42	B	COOP OBSERVER	P	RECUR	OBSERV	Y	S	F
2372	42	B	RECURRING PAYMENT (NO OBLIGATION RECORD)	P	RECUR	SERV	Y	S	F
356	43	A	MULTIPLE PAYMENT PURCHASE ORDER	U	POMULT		Y	P	Y
246	47	B	FAMILY SEPARATION ALLOWANCE	P	NOMTCH	NF5615	Y	S	Y
17	48	B	TORT CLAIM	P	NOMTCH	SF1145	Y	S	Y
170	48	B	EMPLOYEE CLAIM FOR LOSS OF OR DAMAGE TO	P	NOMTCH	CD224	Y	S	Y
66	48	B	PERSONAL PROPERTY CLAIM	P	NOMTCH	SF1034	Y	S	Y

TRANS _NO	FIMA_ DOCUMENT_ TYPE	FIMA_ DOC_ CLASS	FIMA_DOCUMENT_DESCR	SUB SYSTEM_ ID	CFS_DOC1 _CODE	CFS_DOC2 _CODE	ACTIVE_ STATUS	FIMA_2_ REF_FLAG	FIMA_2_ CONTROL _FLAG
352	50	B	TRAVEL ADVANCE APPLIED	P	TRAVNM	ADVAPP	Y	I	F

Appendix F – Issue Log

Issue #	Description / Resolution	Identified By / Assigned To	Status	Opened Date / Closed Date
1.	It is unclear what data and at what level data will be converted from FIMA to CAMS at the time of the cutover.	M. Kirwan	Open (as of 3/21/02)	16-Jan-02
2.	The Account Receivable module of CAMS is still under development and has not been accepted/implemented by NOAA. Will any of this data be available for the Commitment Tracking Interface?	M. Kirwan	Open (as of 3/21/02)	16-Jan-02
3.	The task to implement Summary Level Transfers is still under development and has not been accepted/implemented by NOAA. Will any of this data be available for the Commitment Tracking Interface?	M. Kirwan	Open (as of 3/21/02)	16-Jan-02
4.	The process for applying Labor Estimates has not been tested/implemented in CAMS by NOAA. Will any of this data be available for the Commitment Tracking Interface?	M. Kirwan	Open (as of 3/21/02)	16-Jan-02
5.	The ASAP interface to handle grants is still under development and has not been accepted/implemented by NOAA. Will any of this data be available for the Commitment Tracking Interface?	M. Kirwan	Open (as of 3/21/02)	16-Jan-02
6.	The budget tables in CFS store labor quantities as FTE amounts. The actual labor data provided by the NFC reflects hours. The NOAA Data Mart will need to compute FTE’s from hours and possible hours from FTE’s in order to provide the ability for comparison reporting. NOAA must provide the formula for this calculation.	M. Kirwan	Open (as of 3/21/02)	06-Mar-02

DRAFT Version of Chart

Appendix G – CFS-FIMA Accounts Payable Document Crosswalk (DRAFT)

The following is a copy of the CFS-FIMA Accounts Payable Document Crosswalk that was provided on March 13, 2002. A new version is being developed and will be provided as soon as it is available.

DOCUMENT	**INFORMATION**	**FIMA**	**INFORMATION**				**CAMS INFORMATION**	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Interagency Agreement	Optional Form 347 – Order for Supplies or Services CD-435 Procurement Request	01	OF-347/CD-435 Document # Example: **1EFM62227**	FM040 Purchase Order Transaction Screen	IAGPO	INTAGR	OF-347/CD-435 Document Number Example: **1EFM62227**	FM040 Oblig # 9940

DRAFT Version of Chart

DOCUMENT	INFORMATION	FIMA	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Blanket Purchase Agreement	CD-404 Supply, Equipment, or Service Order	02	BPA Document Number Example: **45JGF8014**	PM003 Invoice Transaction Screen	NOMTCH	SUPPLY SERV	BPA Document # Example: **45JGF8014**	PM003 Trans # 581775
Fedex	Express Small Package Service (ESPS) Interface	02	CAMS PM003 Transaction # + P Example: **000621705P**	PM003 Invoice Transaction Screen	NOMTCH	FEDEX	Fedex Bill to Account # Example: 108722320	PM003 Trans # **621705**
Express Mail (Package delivery services other than those billed through ESPS ie DHL, UPS)	Vendor's invoice	02	CAMS PM003 Transaction # + P Example: **00585107P**	PM003 Invoice Transaction Screen	NOMTCH	EXMAIL	Account #	PM003 Trans #**585107**

DRAFT Version of Chart

DOCUMENT	**INFORMATIO N**	**FIM A**	**INFORMATION**				**CAMS INFORMATION**	
Document Description	**Source Document**	**FIM A Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Documen t Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Non Recurring Contracts	SF-26 Award/Contract	03	SF-26 Document Number Example: **1DKNA9017**	FM040 Purchase Order Transaction Screen	CONTRT	GOODS SERV SUPPLY	Contract # from SF-26 Example: **1DKNA9017**	FM040 Oblig # 9995

DRAFT Version of Chart

DOCUMENT	INFORMATIO N	FIM A	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIM A Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Documen t Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Intergovt Personnel Agreements	OF-69 Assignment Agreement	03	FM040 Oblig # + U Example: **00017559U**	FM040 Purchase Order Transaction Screen	IPA	IPAAGR	IPA Agreement # Example: **2IPA00DRJ**	FM040 Oblig # **17559**
Grants[4]	*Interface* *CD-435 Procurement Request* *CD-450 Financial Assistance Award* *CD-451 Amendment to Financial Assistance Award*	*04*	*FM041 Source Reference* *Example:* ***OA6AC0369***	*FM041*	*GRANT* *ZGRANT* *BGRANT* *YGRANT*	*GRANT*	*Grant Identifying Number in FM041 Source Reference:* *Example:* ***NA06AC0369***	*FM041 Obligation Number*

[4] Grants - CAMS Implementation Date Spring 2002 - Values may be subject to change.

DRAFT Version of Chart

DOCUMENT	INFORMATION	FIMA	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Recurring Deliveries (Non Temporary Storage of Household Goods not related to PCS moves)	Form DD-1164 – Service Order for Household Goods Standard Form 1113 – Public Voucher for Transportation	05	FM040 Source Ref Example: **2AAE0005T**	FM040 Purchase Order Transaction Screen	RECDEL	STORAG	DD-1164/SF1113 Document # Example: **2AAE005TT**	FM040 Oblig # 15246
Work Orders	Optional Form 347 -Order for Supplies or Services	05	FM040 Source Ref Example: **2WW200002**	FM040 Purchase Order Transaction Screen	RECDEL	SUPPLY	Work Order # Example: **2WW200002**	

DRAFT Version of Chart

DOCUMENT	INFORMATION	FIMA	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Lease	SF-2 U S Govt Lease for Real Property GSA Form 276 Supplemental Lease Agreement	06	PM003 Source Reference Example: **4MD0289ZZ**	PM020 Recurring Invoice Maintenance Screen PM003 Vendor Invoice Transaction Screen	RECUR	LEASE	SF-2 or GSA Form 276 Document # Example: **4MD0289ZZ**	PM003 Trans #588582

DRAFT Version of Chart

DOCUMENT	INFORMATION	FIMA	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Utilities	Vendor's Invoice	06	PM003 Source Ref Example: **TE8513700**	PM020 Recurring Invoice Maintenance Screen PM003 Vendor Invoice Transaction Screen	RECUR	UTIL	Utility Account # Example: **TE8513700**	PM003 Trans #629462
GSA Telecommunications [5]	*Interface NOAA Form 37-1 Telecom Service Authorization*	*06*	*PM003 Source Reference* *Example:* ***8N0018798***	*PM003*	*NOMTCH*	*TBD*		*PM003 Trans#*

[5] GSA Telecommunications Interface CAMS Implementation Date FY03

DRAFT Version of Chart

DOCUMENT	INFORMATIO N	FIM A	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIM A Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Documen t Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
GSA Reimbursable Work Authorization	GSA Form-2957 Reimbursable Work Authorization	07	FM040 Source Ref Example: **2N2481096**	FM040 Purchase Order Transaction Screen	GSARWA	RWA	GSA Form 2957 Document # Example: **2N2481096**	FM040 Oblig #11947
Training Authorization	SF-182 – Request, Authorization, Agreement and Certification of Training	08	FM040 Source Ref Example: **2FAKC0003**	FM040 Purchase Order Transaction Screen	TRAIN	TRAIN	SF-182 Document # Example: **2FAKC0003**	FM040 Oblig #16685
Printing[6]	*SF-1* *CD-10*	*09*						

[6] Printing - CAMS Implementation Date Spring 2002

DRAFT Version of Chart

DOCUMENT	**INFORMATIO N**	**FIM A**	**INFORMATION**				**CAMS INFORMATION**	
Document Description	**Source Document**	**FIM A Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Documen t Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Purchase Order/Voucher/ Invoice	SF-44 Purchase Order/Voucher/ Invoice	12	PM003 Source Ref Example: **000188968**	PM003 Vendor Invoice Transaction Screen	NOMTCH	SF44	SF-44 Document # Example: **000188968**	PM003 Trans #650997
Fedstrip/Milstrip (MASC only)	GSA Form 952 – Single Line Item Billing Register SF1080 – Voucher for Transfers Between Appropriations and/or Funds	13	FM040 Source Ref Example: **005AX2003** PM003 Source Ref Example: **031AP1353**	FM040 Purchase Order Transaction Screen or PM003 Vendor Invoice Transaction Screen	FEDSTR or OPAC	FEDSTR FEDSTR	Fedstrip/Milstrip Document # Example: **005AX2003** Example: **031AP1353**	FM040 Oblig#18874 or PM003 Trans#607533

DRAFT Version of Chart

DOCUMENT	**INFORMATION**	**FIMA**	**INFORMATION**				**CAMS INFORMATION**	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Purchase Card	CPCS Interface	14	PM003 Source Ref Example: **PURCHEASC**	PM003 Vendor Invoice Transaction Screen	NOMTCH	PCARD	PURCH + ASC designator Example: **PURCHEASC**	PM003 Trans #626413
Advertising Order	SF-1143 Advertising Order Invoice	19	PM003 Trans # + P Example: **00634262P**	PM003 Vendor Invoice Transaction Screen	NOMTCH	SF1143	SF-1143 Document Ref HEIDI SMITH	PM003 Trans #**634262**
Arbitration Fee* No Recent Examples Available	CD-435 Purchase Request	19	PM003 Trans # + P	PM003	NOMTCH	ARBFEE	Comptroller General's decision #	PM003 Trans #

DRAFT Version of Chart

DOCUMENT	**INFORMATION**	**FIMA**	**INFORMATION**				**CAMS INFORMATION**	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Miscellaneous Invoice	Vendor's Invoice	19	PM003 Trans # + P Example: **00633618P**	PM003	NOMTCH	MISC	Optional Additional Info Example: **002SEA007**	PM003 Trans #**633618**
Travel (TDY Domestic)	Travel Manager Interface	20	PM003 Invoice # Example: **2AN1S0075**	PM003	TRAVNM	DOM	Travel Voucher # Example: TV**2AN1S0075**	PM003 Trans #632430
Travel (GTA Domestic)	GTA Interface	20	PM003 Source Ref Example: **2WT5S1002**	PM003	NOMTCH	GTADOM	Travel Order # Example: **2WT5S1002**	PM003 Trans #519890

DRAFT Version of Chart

DOCUMENT	INFORMATION	FIMA	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Government Bill of Lading (GBL)	Standard Form 1103 – U.S. Government Bill of Lading Standard Form 1113 – Public Voucher for Transportation Charges	23	FM040 Source Ref Example: **0D3851890**	FM040	TRANSP	GBL	SF1103/1113 Document # Example: **0D3851890**	FM040 Oblig #17596
PCS (GTA)	GTA Interface	26	PM003 Source Ref Example: **2AN1P0083**	PM003	NOMTCH	GTAPCS	Travel Order # Example: **2AN1P0083**	PM003 Trans #650907

DRAFT Version of Chart

DOCUMENT	INFORMATION	FIMA	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Travel (TDY Foreign)	Interface	27	PM003 Invoice # Example: **2NA0F2027**	PM003	TRAVNM	FOR	Travel Voucher # Example: TV**2NA0F2027**	PM003 Trans #650298
Travel (GTA Foreign)	Interface	27	PM003 Source Ref Example: **2NS4F0500**	PM003	NOMTCH	GTAFOR	Travel Order # Example: 2NS4F0500	PM003 Trans #520173
GSA Motorpool (CASC only)	Interface	29	PM003 Source Ref Example: **G4137591**	PM003	OPAC	MTRPL	Vehicle Tag # Example: **G4137591**	PM003 Trans #612788

DRAFT Version of Chart

DOCUMENT	**INFORMATION**	**FIMA**	**INFORMATION**				**CAMS INFORMATION**	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Cash In Your Account	CD-326 Recommendation for Recognition	30	PM002 TIN # (Employee's SSN) Example: **18644XXXX**	PM003	NOMTCH	CD326	Awarding NOAA organization Example: DOCNOAANWSNCEP *Note: Employee's SSN goes back to FIMA as document #	PM003 Trans #517724
Imprest Fund Purchase (Direct Reimbursement to Employee)	NF-34-6 Imprest Fund Reimbursement	30	PM003 Trans # +P Example: **00620518P**	PM003	NOMTCH	NF34-6	NOAA organization of requestor Example: **DOCNOAANMFS**	PM003 Trans #**620518**

DRAFT Version of Chart

DOCUMENT	INFORMATION	FIMA	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Local Travel Voucher	TM Interface or paper SF-1164 Claim for Reimbursement for Expenditures on Official Business	31	PM003 Trans # + P Example: **00601891P**	PM003	NOMTCH	SF1164	SF-1164 Voucher # Example: LV2AD0S0012	PM003 Trans #**601891**
Contract Observer	WS Form B66 Payroll Sheet for Contract Observers	33	PM003 Source Ref Example: **0WD000007**	PM003	NOMTCH	OBSERV	Contract Observer Document # Example: **0WD000007**	PM003 Trans #624491

DRAFT Version of Chart

DOCUMENT	INFORMATION	FIMA	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Foreign Trainee Stipends	CD-435 Purchase Request SF-1034 Public Voucher for Purchases and Services other than Personal	35	FM040 Oblig # + U Example: **00015536U**	FM040	FORTRN	STIPEN TRAIN	Trainee’s Name Example: Rafael Vera	FM040 Oblig #**15536**
Oil Company Credit Card	Invoice	36	PM003 Trans # + P Example: **00627816P**	PM003	NOMTCH	GASCD	Vendor’s Invoice Account # Example: 869901207	PM003 Trans #**627816**

DRAFT Version of Chart

DOCUMENT	INFORMATIO N	FIM A	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIM A Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Documen t Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
CSTARS/SPS Purchase Order	CSTARS Purchase Order SPS CD 404 Supply, Equipment, or Service Order	40	FM040 Source Reference Example: **DG2SE0024**	FM040	PO2ADV PO2WAY PO3WAY	EQUIP EXCISE FR NMERC SERV SUPPLY	CSTARS/SPS Document # (9 digit FIMA version) Example: **DG2SE0024**	FM040 Oblig#18760
One-Time Purchase Order (CSPS)	CD-404 Supply, Equipment, or Service Order	40	FM041 Oblig # + U Example: **00013716U**	FM041 CSPS	PO	EQUIP EXCISE FR GOODS NMERC SERV SUPPLY	Optional	FM041 Oblig #**13716**

DRAFT Version of Chart

DOCUMENT	INFORMATIO N	FIM A	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIM A Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Documen t Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Advance Payment PO (CSPS)	CD-404 Supply, Equipment, or Service Order	41	FM041 Oblig # + U Example: **00010355U**	FM041	POAVPY	GOODS SERV SUPPLY	Optional	FM041 Oblig #**10355**
Recurring Payment PO (CSPS)	CD-404 Supply, Equipment, or Service Order	42	FM041 Oblig # + U Example: **00013931U**	FM041	PORCUR	GOODS SERV	Optional	FM041 Oblig #**13931**
Co-op Observers	CD-404 Supply, Equipment, or Service Order	42	PM003 Source Ref Example: **3SW2W0060**	PM020 PM003	RECUR	OBSERV	CD-404 Document # (9 digit FIMA version) Example: **3SW2W0060**	PM003 Trans #637050

DRAFT Version of Chart

DOCUMENT	INFORMATION	FIMA	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIMA Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Document Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Multiple Payment Purchase Order (CSPS)	CD-404 Supply, Equipment, or Service Order	43	FM041 Oblig # + U Example: **00014211U**	FM041	POMULT	EQUIP EXCISE FR GOODS NMERC SERV SUPPLY	Optional	FM041 Oblig #**14211**
Family Separation Allowance	NOAA FORM 56-15 Family Separation Allowance Claim & Authorization	47	PM003 Trans # + P Example: **00424809P**	PM003	NOMTCH	NF5615	NOAA Corps Officer's Name Example: HAGANJEFF	PM003 Trans # **424809**

DRAFT Version of Chart

DOCUMENT	INFORMATIO N	FIM A	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIM A Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Documen t Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Personal Property Claim/Tort Claim Voucher	Standard Form 1145 – Voucher for Payment Under Federal Tort Claims Act Standard Form 1034 – Public voucher for Purchases and Services other than Personal Form CD-224 – Employee Claim for Loss of or Damage to Personal Property	48	PM003 Trans # + P Example: **00629079P**	PM003	NOMTCH	SF1145 SF1034 CD224	Type of Claim Example: TORT CLAIM	PM003 Trans #**629079**

DRAFT Version of Chart

DOCUMENT	INFORMATIO N	FIM A	INFORMATION				CAMS INFORMATION	
Document Description	**Source Document**	**FIM A Doc Type**	**9-Digit FIMA Document No**	**CAMS Initial Point of Entry**	**Documen t Type**	**Item Types**	**CAMS Source Reference Field Value**	**System Generated CAMS Transaction No**
Travel Advance	Interface CD-369 Travel Advance	50	PM003 Invoice # Example: **2MAPF1019**	PM003	TRAVNM	ADVAPP	Travel Voucher # Example: TV**2MAPF1019**	PM003 - FIMA

Appendix H – Batch Routine Log File Layout

```
                                                                ROUTINE.SQL Log Report

Routine Name:               ROUTINE.SQL
Date/Time Process Started:  mm/dd/yyyy hh:mm:ss AM
REFRESH ID / PASS #:  999999/999
PARAMETERS USED:
NDW_REFRESH_PARAMS.BEGIN_TRANS_NO (KEY FIELD):        999999999
NDWnnn_RECORDS_TO_PROCESS:                            999999999
NDWnnn_RECORDS_TO_COMMIT:                             999999999

PARAMETERS UPDATED:
NDW_REFRESH_PARAMS.BEGIN_TRANS_NO (KEY FIELD):        999999999

NDW_PROCESS_LOG:
DATE/TIME            MODIFICATION DATE      TRANS NO  TABLE NAME            RECORDS     AMOUNT               STEP DESCRIPTION
-------------------  ---------------------- --------  --------------------  ----------- -------------------  -----------------------------------------
mm/dd/yyyy hh:mm:ss  mm/dd/yyyy hh:mm:ss XM 99999999  TRIAL                 999,999,999 $999,999,999,999.99  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Example:
mm/dd/yyyy hh:mm:ss                         99999999  TRIAL                                                  Routine started with TRIAL_ID > this
number.

mm/dd/yyyy hh:mm:ss                                   TRIAL                         100 $        111,111.11  Cumulative summarized records committed
                                                                                                             to NDW_STAGING table - ROWID = 999999.

mm/dd/yyyy hh:mm:ss                                   TRIAL                         200 $        222,222.22  Cumulative summarized records committed
                                                                                                             to NDW_STAGING table - ROWID = 999999.

mm/dd/yyyy hh:mm:ss                                   TRIAL                         300 $        333,333.33  Cumulative summarized records committed
                                                                                                             to NDW_STAGING table - ROWID = 999999.

mm/dd/yyyy hh:mm:ss                         99999999  TRIAL                         325 $        333,444.44  Last TRIAL_ID processed updated in
                                                                                                             NDW_REFRESH_PARAMS table.

mm/dd/yyyy hh:mm:ss                         99999999  TRIAL                 999,999,999 $999,999,999,999.99  Process completed.  Count and control
                                                                                                             total of TRIAL records processed.


Date/Time Process Ended:  mm/dd/yyyy hh:mm:ss AM
```